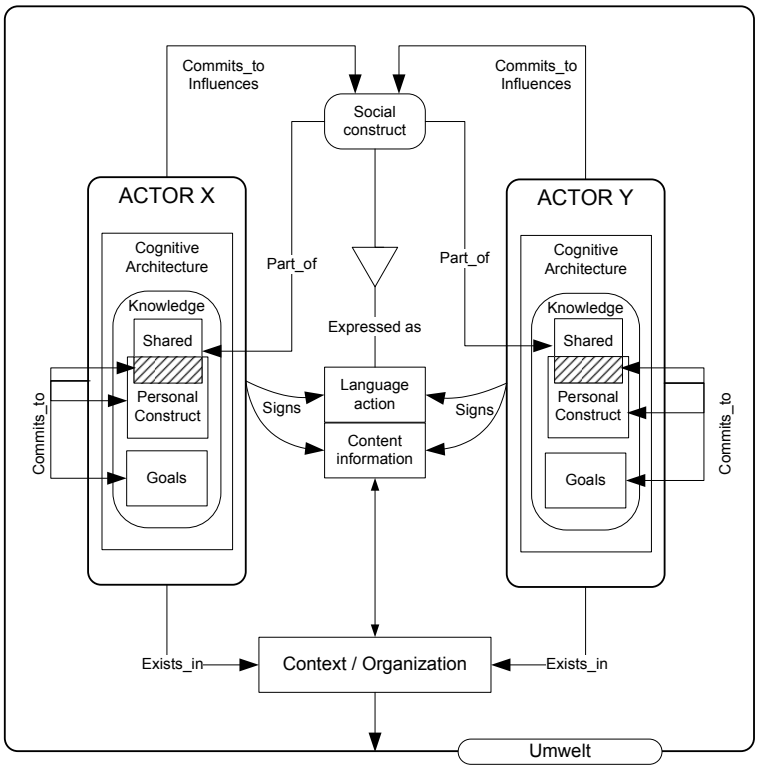
The Social Cognitive Actor
A multi-actor simulation of organisations



MARTIN HELMHOUT

Publisher: Labyrint Publications
Postbus 334
2984 AX Ridderkerk
The Netherlands
Tel: 0180-463962

Printed by:  Offsetdrukkerij Ridderprint B.V., Ridderkerk

ISBN-10: 90-5335-100-0
ISBN-13: 978-90-5335-100-0

This document was prepared using the LaTeX document preparation system.

Rijks*Universiteit* Groningen

The Social Cognitive Actor

A multi-actor simulation of organisations

Proefschrift

ter verkrijging van het doctoraat in de
Bedrijfskunde
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. F. Zwarts,
in het openbaar te verdedigen op
donderdag 30 november 2006
om 13.15 uur

door

Jan Martin Helmhout

geboren op 31 januari 1974
te Assen

Promotores:	Prof. dr. H. W. M. Gazendam Prof. dr. R. J. Jorna
Beoordelingscommissie:	Prof. dr. K. Liu Prof. dr. A. M. Sorge Prof. dr. R. Sun

Preface

The dissertation that is in your hands is the fruit of the last four years of research done in the field of Multi-Agent Systems, Artificial Intelligence, Informatics, Cognitive Science, Social Science and Organisation Studies. Bringing together different disciplines has been a challenging and a complex task. The outcome, put together in a simulation model, will hopefully help others discover new ways of practising the art of cognitive and social modelling.

My background as a mechanical engineer may not seem to be in line with the research that I am currently involved in. Indeed, constructs of the philosophy of the mind, social construction, multi-actor software programs, and social behaviour seem to differ, in essence, from mechanical engineering.

However, this dissertation focuses on the process of construction as in mechanics. Only this time, the focus is on social construction. It is probably the topic of social construction that (re-)constructed my thoughts about the (social) world we live in.

I want to thank my promoters Prof. Henk Gazendam and Prof. René Jorna for this change in looking at the world around me. Although the many discussions I had about the functional structure of the mind and how social constructs, shared by people, influence behaviour even at the lower level of thinking that complicated my vision of the world, they also challenged me to the extent of developing a workable model.

The role of a PhD candidate is a social construction in its own right, created by a mutual agreement between the University of Groningen, the SOM research school, my promoters and I. I want to thank the University of Groningen, SOM and my promoters for giving me the opportunity to engage in this social construction.

Many people played an important role in contributing to the work delivered in this dissertation. In the early phase of my PhD, I worked together with Diederik Kraaikamp, Luit Gazendam, René Jorna, Tomas Klos, Jos Schreinemakers and Bart Nooteboom on Agent-Based Computational Transaction Cost Economics. I thank all of them for the fruitful discussions I had with them that delivered new ideas and experience of how a cognitive (social) actor should be modelled (see Kraaikamp, 2003).

Next, a period of programming started that resulted in a software package which can be downloaded from http://sourceforge.net/projects/act-rbot. The software programming consumed many hours of dedication not only by me but also Gijs Roest. Our joined efforts resulted in the RBot-Project (see Roest, 2004) in which Gijs concentrated at creating a new memory model that incorporated the ACT-R model successfully. Gijs, thank you for being such a critical and inventive colleague.

The people who probably understand my work the best are Henk Gazendam and René Jorna. Henk Gazendam is to be admired, because he not only reviewed my theoretical work, but also my software by transforming his SmallTalk skills into JAVA. He is to be complimented on assimilating many classes and lines of software code. Henk was a constructive reviewer. He not only provided comments like 'onzin' (nonsense) on my work, but also shared ideas and gave detailed arguments that helped improve my work tremendously. Henk, thank you very much for your detailed feedback.

My second supervisor René is an intelligent and very active researcher who is interested in the cognitive actor; according to René, an actor is *not* an (economic) agent. As a researcher and a cognitive psychologist, he managed to open up the world of cognition to me. As a managerial actor, René always tried to find a way to make things possible, be it for research purposes or others, and I have to admit, he always found a way. René, thank you for sharing your cognitive skills and your passion for research with me.

I also want to thank my promotion committee: Prof. Kecheng Liu as an organisational semiotician and applied informatics expert, Prof. Arndt Sorge as an organisational sociologist and Prof. Ron Sun as a cognitive scientist and cognitive multi-agent modeller. I am grateful to them for their suggestions, time and the careful reading of my manuscript. I also would like to thank Prof. John Michon who as a cognitive scientist and Prof. Ron Stamper who as an expert in organisational semiotics provided valuable insights.

Furthermore, I want to thank all the PhD students and people with whom I shared experiences, office space and the complex of WSN and DRS building during my PhD years: Niels, Kristian, Nanda, Marloes, Gerben, Laura, Joost,

Larissa, Ruben, Jan, Filippo, Jacques, Thomas, Bert, Herman, Simon, Hans, Hans, Gert, Gijs, Nick, Mark, Petra, Xander, Sonja, Wander, Kees, Frank, Johanna, Roger, Jan, Willem, Derk-Jan, Marjolein, Wim, Marina and of course the secretaries Sonja, Hennie, Monique, Tanja, Annet and Durkje.

A special thank you goes out to Niels, one of my paranimfen, who always was in for discussions, solved programme puzzles and was a great support throughout my PhD years.

Next, I want to thank my friends in Utrecht and Heerenveen, as well as my family. In particular, I want to dedicate this work to my uncle, a pure scientist (a physicist), who loved academic work as much as I do. I would also like to thank my mother Jannie, Jans, Hanita, Rolf, Mark and Frank who had to bear the burden of listening to many stories about the challenges of doing a PhD. Besides that I want to thank my family in law Polat, Gülten, Banu, Alper, Ilay, Çigdem and Ali.

And finally, I want to thank my wife Ayşe. Ayşe gave me so much, not only her love and care, but also academic support and the consistency of her bright mind.

Groningen, 24 October 2006

Martin Helmhout

To the past and the future,
to my father and son, Anne.

Contents

List of Figures **v**

List of Tables **ix**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Research background . . . 4
1.3 Methodology . . . 6
1.3.1 The design-based approach . . . 8
1.3.2 Life cycle of a simulation study . . . 9
1.4 Research questions and objectives . . . 11
1.5 Dissertation outline . . . 14

2 Multi-Agent Systems **17**
2.1 Agency . . . 21
2.1.1 Weak notion of agency . . . 22
2.1.2 Strong notion of agency . . . 22
2.2 Agent components and architecture . . . 24
2.2.1 Perception and attention . . . 24
2.2.2 Action . . . 25
2.2.3 Interaction between perception and action . . . 25
2.2.4 Cognition . . . 28
2.2.4.1 Memory . . . 28
2.2.4.2 Learning . . . 29
2.2.4.3 Goal-directed behaviour, meta-cognition and emotion . . . 31
2.2.4.4 Environment, communication, language and social ability . . . 32
2.3 The cognitive and the social agent . . . 33
2.3.1 The reflexive and reactive agent . . . 34

2.3.2 The reasoning agent . . . 34
2.3.3 The reflective agent . . . 35
2.3.4 The social agent . . . 36
2.3.5 The complete cognitive and social agent . . . 37
2.4 The environment of the agent . . . 42
2.5 Agent and Coordination . . . 45
2.5.1 Cooperation and Conflict . . . 45
2.5.2 Coordination mechanisms . . . 48
2.5.2.1 The economic perspective . . . 49
2.5.2.2 The DAI perspective . . . 50
2.6 Multi-Agent System Applications . . . 58
2.6.1 Multi-Agent decision systems . . . 59
2.6.2 Multi-Agent simulation systems . . . 60
2.7 Discussion . . . 60

3 The Social Actor **63**
3.1 Social constructivism . . . 65
3.2 Social construction of organisation and society . . . 69
3.3 Social Environment and Semiotics . . . 71
3.3.1 Umwelt . . . 71
3.3.2 Communication and Semiotics . . . 74
3.4 Semiotic level, social constructs and the social actor . . . 83
3.4.1 The situated actor and the semiotic level . . . 83
3.4.2 Social constructs . . . 85
3.4.3 Social constructs and organisation . . . 90
3.4.4 Social constructs: creation, evolution, control and ending . 93
3.5 Discussion . . . 97

4 The Cognitive Actor **101**
4.1 Cognitive science . . . 102
4.1.1 Levels of description . . . 103
4.1.2 Connectionism and embodied cognition . . . 105
4.2 The Cognitive Architecture . . . 109
4.2.1 SOAR and ACT-R . . . 110
4.2.1.1 Summary of SOAR . . . 110
4.2.1.2 Summary of ACT-R . . . 112
4.3 Comparing SOAR and ACT-R: selecting a cognitive architecture . 113
4.3.1 Selection of a cognitive architecture . . . 115
4.4 ACT-R . . . 121
4.4.1 Perception and Motor module . . . 123
4.4.2 Goal module . . . 124
4.4.3 Declarative module . . . 125
4.4.3.1 Activation, base-level learning and context of chunks . . . 125
4.4.3.2 Retrieval parameters of declarative chunks . . . 130
4.4.3.3 Declarative chunks and learning: two types of learning . . . 132

4.4.4 Procedural module . . . 133
4.4.4.1 Procedures and symbolic procedural learning . . 136
4.4.4.2 Procedures and sub-symbolic procedural learning 138
4.4.5 The cognitive engine . . . 143
4.4.5.1 Matching (Step 1) . . . 145
4.4.5.2 Conflict Resolution (Step 2) . . . 145
4.4.5.3 Execution (Step 3) . . . 147
4.4.5.4 Credit assignment (Step 4) . . . 147
4.4.5.5 Check end-state (Step 5) . . . 149
4.5 Evaluation of ACT-R . . . 149
4.5.1 Multi-Agent System . . . 149
4.5.2 Social behaviour: social interaction and social constructs . 151
4.5.3 The rational and functional level . . . 153
4.6 Discussion . . . 156

5 RBot: Architecture and Design 159
5.1 General Requirements and Overview of the Architecture . . . 160
5.1.1 Patterns . . . 161
5.1.2 Overview of the architecture . . . 162
5.2 The cognitive actor architecture . . . 166
5.2.1 The Memory . . . 168
5.2.1.1 Memorymap, buffer and stack . . . 169
5.2.1.2 Memory Components . . . 171
5.2.2 Controllers and handlers . . . 181
5.2.2.1 Goal Feeder and Goal Handler . . . 182
5.2.2.2 Perception . . . 188
5.2.2.3 Action :: physical memorymap and position handler . . . 189
5.2.2.4 Action :: message memorymap and message handler . . . 190
5.2.2.5 Memory components and estimating the sub-symbolic properties . . . 191
5.2.2.6 Social construct handler . . . 191
5.2.3 View and control of the model . . . 193
5.3 Backend server . . . 194
5.4 The Mixed-Tier . . . 195
5.4.1 Server: control, registration, synchronisation and communication . . . 196
5.4.1.1 Control of the simulation task environment . . . 196
5.4.1.2 Registration of actors . . . 196
5.4.1.3 Synchronisation of processes . . . 196
5.4.1.4 Communication among actors & between actors and task environment . . . 200
5.4.2 The task environment . . . 202
5.4.3 The client: graphs and data presentation . . . 205
5.5 Discussion . . . 206

6 Demonstrative experiments with RBot and MRS **209**
6.1 Verification, Validation and Testing 209
6.1.1 Validation methods and techniques 213
6.1.2 The combined cycles of RBot and MRS 215
6.2 Demonstrative experiments 215
6.2.1 Experiment 1: RBot vs. ACT-R 216
6.2.1.1 The addition-problem 216
6.2.1.2 Results of the addition-problem experiment . . . 219
6.2.1.3 Discussion 222
6.2.2 Experiment 2: Emergence of Social Constructs and Organisational Behaviour 222
6.2.2.1 Description of the experiment 223
6.2.2.2 The model of the experiment 225
6.2.2.3 Results 226
6.2.2.4 Discussion 229
6.2.3 Experiment 3: Social construct as coordination mechanism 231
6.2.3.1 Description of the experiment 232
6.2.3.2 Results 234
6.2.3.3 Discussion 236
6.3 Discussion and Conclusion 237

7 Discussion and Conclusions **241**
7.1 Research findings 241
7.1.1 Theoretical conclusions 244
7.1.2 Implementation and experimental conclusions 247
7.2 New questions raised 250
7.2.1 Questions concerning RBot 251
7.2.2 Questions concerning MRS 253
7.2.3 Questions concerning simulation and application 254
7.3 Further work 255

A Apendix: Addition Productions **259**

Bibliography **263**

Author Index **283**

Index **289**

Summary **297**

Samenvatting **303**

List of Figures

1.1 The logic of simulation as a method. 7
1.2 Life Cycle of a Simulation Study. 10
1.3 Structure of the dissertation. 15

2.1 History of contemporary approaches of (social) simulation theories and technique. 17
2.2 Taxonomy of memory types. 28
2.3 Agent with components. 33
2.4 Nesting of environments. 43
2.5 Typology of coordination relationships. 45
2.6 The three states of CDPS. 51
2.7 Causal network for commitment to joint goal. 54
2.8 ARCHON (GRATE*) agent architecture. 55

3.1 Social constructivism and overview of levels and theories. 65
3.2 Functional circle. 73
3.3 Communication model. 77
3.4 Components of the actor. 80
3.5 Social constructs present at three levels. 84
3.6 Social constructs, actors and context. 91
3.7 Four meanings of a social construct. 99

4.1 The subsumption architecture. 108
4.2 Relationship between theory, architecture, models, actor and cognition. 110
4.3 Architectural structure of SOAR. 112
4.4 The organisation of information in ACT-R 5.0. Information in the buffers associated with modules is responded to and changed by production rules. 123
4.5 Chunk addition-fact 7. 125

4.6 Example chunk, accessed at times 10 and 28. 127
4.7 Activation spread between goal chunk and three chunks in memory. 128
4.8 Procedure pattern. 134
4.9 Example of a procedure pattern. 134
4.10 ACT-R engine and interaction. 145
4.11 Conflict resolution / credit assignment. 146

5.1 Patterns for component interaction and distribution. 163
5.2 Tier client-server architecture; Multi-RBot System (MRS). 164
5.3 Example of a MVC pattern. 166
5.4 RBot; comparable with ACT-R. 168
5.5 Three types of memorymaps. 170
5.6 Example of a buffer / memorymap. 171
5.7 Example of a memory structure. 173
5.8 Memory components class diagram. 174
5.9 Structure of the procedure. 176
5.10 Classes that implement the social construct. 181
5.11 GoalFeeder & finite state machine of the GoalHandler. 183
5.12 Goalstack. 185
5.13 Positionhandler. 189
5.14 Social constructs and the impact on utility of procedures. 192
5.15 User interface of RBot: view and control. 193
5.16 Possible database connection models. 195
5.17 Registration of an actor. 197
5.18 The next-event time-advance approach. 197
5.19 The fixed-increment time advance. 198
5.20 Multi-actor synchronisation clock. 199
5.21 Timer communication mechanism. 199
5.22 A task environment in action. 203
5.23 Server controls who perceives what. 203
5.24 Communication between actor and environment. 204
5.25 User interface that presents data from the database in a graph. . . 206

6.1 Life Cycle of a Simulation Study; see also chapter 1. 211
6.2 The two development cycles of RBot and MRS. 215
6.3 A multi-column addition problem. 218
6.4 Activation: base level learning and retrievals of chunk 1+1 = 2 and chunk 5+ 7 = 12. 220
6.5 Creation of a chunk and the merger of a chunk. 221
6.6 Norm selection as a coordination game. 224
6.7 Nash equilibrium development over time. 224
6.8 Game Frame of two actors trying to pass each other. 226
6.9 States and state changes of the productions of an actor. 226
6.10 Actor1 & 2: Utility, Successes, Failures of procedure 'Left' & 'Right'. 228
(a) Actor1: Successes, Failures Left 228
(b) Actor1: Successes, Failures Right 228
(c) Actor1: Utility Left 228

(d) Actor1: Utility Right 228
(e) Actor2: Successes, Failures Left 228
(f) Actor2: Successes, Failures Right 228
(g) Actor2: Utility Left 228
(h) Actor2: Utility Right 228
6.11 Y coordinates of both actors per time step. 229
6.12 The extension of the cognitive engine with social constructs. 232
6.13 Actor1 & 2: Utility, Successes, Failures of procedure 'Left' & 'Right'. 235
(a) Actor1: Successes, Failures Left 235
(b) Actor1: Successes, Failures Right 235
(c) Actor1: Utility Left 235
(d) Actor1: Utility Right 235
(e) Actor2: Successes, Failures Left 235
(f) Actor2: Successes, Failures Right 235
(g) Actor2: Utility Left 235
(h) Actor2: Utility Right 235
6.14 Y coordinates of both actors per timestep. 236

List of Tables

2.1 Characteristics of the social and cognitive agent. 41
2.2 Classification of interaction scenarios. 47
2.3 Strategic form of game theory; the prisoner's dilemma. 49

3.1 Jakobson's six functions of languages. 76
3.2 Explanations of communicative functions. 78
3.3 Similarity between two approaches of communication. 80
3.4 KQML dialogue. 82

4.1 Levels of description of various cognitive scientists. 103
4.2 Comparison of SOAR and ACT-R. 118
4.3 Time scale of human action. 151

5.1 Example of a procedure and a goalchunk in XML. 179
5.2 Format of message chunk. 190
5.3 General markup of the actor message. 201
5.4 Content of RBot message. 201

6.1 Operational Validity Classification. 214
6.2 Goal stack addition problem. 219
6.3 Social constructs (XML format). 234

CHAPTER 1

Introduction

MULTI-AGENT SYSTEMS (MAS) is a field of research that is concerned with the design, analysis and implementation of computational decision and simulation systems—a collection of agents that work in conjunction with each other—to enhance one's understanding of how people cooperate and coordinate their actions to solve problems. Upon its foundation in the mid-nineties, MAS propagated in the fields of cognitive science and social science, and was adopted, more specifically, in cognitive modelling and modelling of multi-agent interaction as in, for instance, social simulation (Conte & Gilbert, 1995; Troitzsch, 1997). However, there has not been enough interaction between these two fields, and tools that try to connect both fields have not been sufficiently developed (Sun, 2006b). In our research, we will address this problem and make an attempt to bring both fields—cognitive science and social science—closer together.

This chapter is structured as follows. Section 1.1 discusses the motivation for selecting organisational, social and cognitive theories and the need for the development of a cognitive agent-based computational social simulation model. Section 1.2 presents the research background of the dissertation. In section 1.3, we will elaborate on the methodology applied in this research. Next, the research questions and objectives are outlined in section 1.4 and we end this chapter with section 1.5 that provides an outline of the dissertation.

1.1 Motivation

Social sciences have adopted MAS for the creation of social simulations in order to shed light on organisational and sociological phenomena. However, as Castelfranchi (2001) points out, a significant theoretical problem exists in the field of social sciences; there is a lack of understanding or explanation of unconscious, unplanned forms of cooperation among intentional agents. Moreover,

Moss (2006) states that science that eschews observation in favour of formalism is rife in social science. His claim is that agent-based models can capture independent validation by "...ensuring that the specification is well verified with respect to formal models from cognitive science understanding that those formal models are themselves well validated experimentally and observationally" (ibid., p. 398).

According to Castelfranchi (2001), Artificial Intelligence (AI) and, in particular, MAS—the cognitive modelling of agents and its learning aspects—can contribute to the simulation of *artificial societies*[1]. On the other hand, cognitive science needs Multi-Agent Systems, social simulation, and social sciences in general (Sun, 2006b). There is a real need for new complex cognitive models that take into account social factors in cognition. In other words, there is a shortage of good ideas and theories that address socio-cultural concepts/signs/symbols within social structures from a cognitive standpoint (Sun, 2001). Cognitive agent-based social simulations have adopted MAS to give insights into social-level phenomena based on the individual agent's actions, i.e. the cognitive level.

The aim of this dissertation is to design and implement a cognitive agent-based computational social simulation model based on a selection of social and cognitive theories in an attempt to satisfy the need for a complex cognitive-social model. In comparison to models emanating from traditional social (and economic) sciences, cognitive-social models enable one to gather, organise, and interpret observations and data more directly and test various factors more thoroughly (Sun, 2006b). Furthermore, such models are validated by incorporating behavioural descriptions and previously validated concepts from cognitive science and social psychology at the micro level, and by capturing the statistical output of (longitudinal) studies in the field of social science at the macro level (Moss, 2006).

The motivation for this research is twofold: (1) to reveal the constituents of MAS and the social cognitive agent (cf. Conte & Castelfranchi, 1995a) based on theoretical considerations to (2) construct a simulation model that plausibly explains the interactive social behaviour of agents in a physical and socially situated environment (cf. Gilbert & Troitzsch, 1999). The aim is to relate the behaviour of individuals (micro level) that form a group or an organisation to the behaviour and performance of a group or organisation as a whole (Alexander & Giesen, 1987; Van den Broek, 2001). This rests on the 'methodological individualism' assumption that supports MAS methodology:

> "*all* description and explanation of social phenomena should ultimately be in terms of individuals, their properties, and their interrelations in terms of these properties (Franssen, 1997)" (Van den Broek, 2001, p. 26).

Similarly, it is established in social constructivism (Mead, 1934) that society and all its related issues, such as money and marriage, are products of (symbolic) social interaction between individuals. These interactions are caused by

[1] Artificial societies is used here to refer to social simulations using MAS (Minsky, 1985; Sawyer, 2003).

actions of individuals as part of a group. Accordingly, the individual and its interactions with others are the objects of study in this dissertation. Semiotics, as a study of signs and sign systems, is also addressed in this dissertation and is concerned with the *'symbolic interaction'* of individuals and the (shared) meaning that they assign to constructs (i.e. semiotic codes) which they create through semiosis[2].

In MAS, it is common to define varying *levels of description*; for instance from the biological, cognitive, rational to the social level of description (Newell, 1990). Social and economic sciences often operate at the social level. This is acceptable so far as the properties of actors at lower levels are assumed to be constant (Gilbert, 2006). However, when the analysis of lower levels constrains that at higher levels, or when there is a need to model the social and cognitive levels, then a 'mixed-level' analysis is necessary (Sun, 2006b). Under that circumstance, descriptions at one level can be applied to other levels (Gilbert, 2006).

In this dissertation, a mixed-level analysis is performed, in which the following levels are discerned in descending order:

1. The *social level* is involved with social laws and overall behaviour of the social system. The advantage of such a level "is that it enables the overall system's behaviour to be studied without having to delve into the implementation of the individual [actors]" (Jennings & Campos, 1997, p. 3), e.g. it can be explained with the help of population statistics or social network characteristics. However, we argue, as this dissertation will make clear, that the individual and other levels of description are important for a better explanation, understanding and prediction of social behaviour (methodological individualism).

2. The *semiotic level*[3], which describes the use of language and signs in communication, interaction and negotiation in order to agree on social constructs (e.g. common plans, or contracts) (Gazendam, 2004; Helmhout et al., 2004; Helmhout, Gazendam, & Jorna, 2005b).

3. The *intentional level*, which ascribes beliefs, desires and intentions to actors. Intentions of others, inferred from knowledge about others' beliefs and desires, enable the examination of others' actions (Dennett, 1987).

4. The *functional level* describes learning and cognitive mechanisms of the actor and is grounded in an empirically validated theory (cf. Anderson & Lebiere, 1998).

5. The *physical/physiological level* is a level described by appealing to physics, biology and chemistry. It predicts or explains behaviour in terms of physical laws or physiological properties (Dennett, 1987).

[2]Semiosis encompasses the triadic notion of meaning processes, i.e. the interaction between representamen, the object and the interpretant (Peirce, 1931).

[3]Actually, semiotics can be assigned to the other levels as well. For instance in cognitive science, the symbol system hypothesis is used for descriptions at the functional level. The semiotic level is distinguished, because organisational semiotics focuses on this level in which social constructs, speech acts, interaction scenarios, shared knowledge units, and so on can be defined (Helmhout, Gazendam, & Jorna, 2004).

A complete design and implementation of a MAS has to incorporate the basic aspects of all the levels listed above. As signs and symbols are common to all levels, MAS and its agents should support the processing of signs and symbols. The implication for the overall system is that:

1. The actor should be able to process and create signs and symbols, i.e. a cognitive plausible agent is a requirement.
2. The environment should support:
   (a) Physical objects that represent signs and symbols.
   (b) A communication and social environment that provides an infrastructure for the transfer of signs and symbols and allows actors to interact in a coordinated manner.
3. The actor must have the capabilities to (socially) construct the world and use a common medium, or language, to exchange, negotiate and reach agreements about (social) constructs with other actors.

A simulation with a MAS can provide (1) valuable insights into and description of organisational behaviour, cognitive and social phenomena (for instance, emergence of organisations), (2) locate errors and gaps in verbal theories, (3) demonstrate which theoretical propositions are logically consistent, (4) predict the state of an agent or system in the future, and (5) discover, formalise and test (new) theories based on the simulation outcomes (Carley, Prietula, & Lin, 1998; Gilbert & Troitzsch, 1999).

In this dissertation, we will provide a selection of theories that are applicable for the design of a MAS. We will construct a model of a MAS, named MRS (Multi-RBot System) and a cognitive and social actor, named RBot, by drawing on these theories. An overview of these theories is provided in the following section.

1.2 Research background

This research draws on many theories or research fields such as organisational behaviour and structures, Computational & Mathematical Organization Theory (CMOT) (Carley & Prietula, 1994a; Carley & Gasser, 1999), organisational semiotics (Gazendam & Liu, 2005), cognitive science/artificial intelligence, social constructivist theory and MAS.

CMOT enables the modelling of business organisations, its organisational and authority structures, organisational skills and procedures depending on the position the agent takes in the organisation (Sawyer, 2003). With the help of CMOT, researchers create a formal model to make predictions about an organisation as an adaptive system composed of agents who themselves are adapting (Carley et al., 1998).

Organisational semiotics tries to understand organisations based on the use of signs, texts, documents, sign-based artefacts and communication, drawing on the basic disciplines like psychology, economics, and information systems

science (Gazendam, 2004). It develops this perspective using the established discipline of semiotics, the theory of signs. The strength of semiotics lies in the theory of significant symbol processing (sense making, interpretation, knowledge, information, and culture) and provides a (systematic) way of looking at issues of meaning, interpretation and knowledge—a theory of human culture (Van Heusden & Jorna, 2001). The adjustment to one another of the acts of different human individuals within the human social process (semiosis) takes place through communication by gestures and significant symbols (Mead, 1934, p. 75). This symbol, clearly present in organisational semiotics, contributes to the explanation of the social and organisational processes and facilitates a deeper understanding of specific aspects related to organisational representation and behaviour.

Apart from the organisational aspect, cognitive science and artificial intelligence and, more specifically, cognitive architectures—the design and organisation of the mind—have been studied. Theories of cognitive architectures strive to provide a unified/generalised theory of cognitive systems, a description of the functions and its capacities and a set of principles for constructing a set of models, rather than a set of hypotheses to be empirically tested (Sloman, 1999). The two streams of architectures that can be distinguished are the symbolic and the connectionist stream of cognitive modelling. The first architecture is based on a production system[4], e.g. SOAR (Lehman, Laird, & Rosenbloom, 2006; Newell, 1990), and the second is based on (neural) networks, in which processing is highly distributed. In contrast to the symbolic architecture, there are no task related modules, discrete symbols, or explicit rules present that govern operations (Rumelhart, McClelland, & PDP Research Group, 1986).

According to Sloman (1996, 1999), hybrid architectures, e.g. ACT-R (Anderson, 1983; Anderson & Lebiere, 1998) and CLARION (Sun, 2003) that combine elements of both streams, are necessary, because symbolism tends to be capable of manipulating variables in a way that matches human competence. On the other hand, connectionism tends to be better in detecting similarities and contiguity[5] as well as people's ability to deal with and integrate many pieces of information simultaneously.

In this dissertation, a hybrid architecture is chosen for modelling the actor in order to create a more realistic basis for understanding the impact of individual behaviour in artificial societies. Apart from this division in cognitive architectures, there is a division in streams within AI. One is the classical approach in which the agent holds a symbol system or internal representation of the environment to which the agent applies rules, frames, semantic nets or logics, to reason about itself and its (social) position in the environment. The other approach is the situated, or behaviour-oriented approach (Brooks, 1991b; Steels, 1994). The situated or behaviour-oriented approach defines intelligence in the form of observed behaviour and without explicit representation of symbols and without abstract reasoning such as symbolic AI. The systems often have a set

[4]Production systems are computer languages that are widely employed for representing the processes that operate in models of cognitive systems (Newell & Simon, 1972; Newell, 1973).

[5]Contiguity refers to pattern recognition, i.e. the sequential occurrence or proximity of stimulus and response, causing their association in the mind (Pearsall, 2002).

of stimulus-response reactions, i.e. depending on the situation a proper action repertoire is chosen. Situated AI is closely connected to the environment and therefore is often referred to as embodied cognition[6]. However, the type of situatedness of embodied cognition is often *physical* and *not* social-cultural. The social constructivist theory fills in the void of limited social-cultural situatedness.

The social constructivist theory (cf. Blumer, 1969; Mead, 1934; Vygotsky, 1962, 1978) can provide the theoretical support for bridging the gap between the behavioural and the classical approach, and introduce social aspects to AI. The social constructivist theory explains how the individual, the mind and the self become part of the society through interaction with others and how the society becomes part of the self[7] (Mead, 1934). There is a growing interest in social situatedness and the social construction of reality (e.g. Brooks & Stein, 1994; Dautenhahn, 1995; Dautenhahn, Ogden, & Quick, 2002; Edmonds, 1998; Lindblom & Ziemke, 2003). Social factors in cognition are being introduced to models to advance their cognitive basis in relation to their social environment, e.g. CLARION (Sun, 2002, 2003).

In this dissertation we also apply theories of Multi-Agent Systems (MAS), which originates from the field of Distributed AI (DAI) (Bond & Gasser, 1988; O'Hare & Jennings, 1996). In DAI, most systems have a centralised control system, but with the birth of Internet and multi-processor platforms, researchers have started to experiment with decentralised control structures, in which every node has its own autonomy[8]. This notion of autonomy marked the start of the current commonly-used term agent or actor; an agent is situated in an environment and is capable of autonomous action in that environment (Wooldridge, 1999).

Multi-Agent Systems deal with autonomous agents that have control over their internal state and the way their actions influence the environment. In this dissertation, we are interested in the (socially constructed) interactions between agents and the organisational aspects as regards coordination and cooperation. The field of MAS also clarifies how models for investigating the behaviour of societies and organisations can be designed and built.

But before building a model, first an appropriate methodology needs to be selected. In the next section, we discuss the methodology that is applied in this dissertation.

1.3 Methodology

As mentioned before, the aim of this dissertation is to develop an instrument or simulation model. The development of our model requires a methodologi-

[6]Embodied Cognition is a growing research program in cognitive science that emphasises the formative role that the environment plays in the development of cognitive processes (Cowart, 2006).

[7]In the interaction with society, the individual internalizes not only the character of the individual but also the roles of all others with whom he has interaction, the so called 'generalized other' (Mead, 1934, p. 154).

[8]Autonomy is an agent's active use of its capabilities to pursue some goal without intervention by another agent in the decision making process that could possibly influence how that goal should be pursued (Barber & Martin, 1999).

cal approach that supports the development of an (simulation) instrument. In general, we can distinguish two approaches to conduct research: the *empirical cycle*—explaining phenomena by developing theory, and the *regulative cycle*—"solving real, existing problems in a well-defined, step-by-step manner using theoretical frameworks and models to derive a solution to the problem under scrutiny"(Homburg, 1999, pp. 22–23)[9].

The following distinction makes the difference and the resemblance between those two approaches clear (see Gilbert & Troitzsch, 1999). Statistical modelling conforms to the empirical cycle in which the researcher develops a model based on theory, a set of hypotheses or equations that results in a measurement model thereby creating an abstraction of the real world. The next step is to find out whether the model generates the predictions by comparing them to data that have been collected in practise (typically assessed by means of tests of statistical hypotheses). In comparison to statistical modelling, the logic of simulation as a method (see figure 1.1) is quite similar. However, in this case, the aim is to develop an instrument[10] whose methodology is similar to that of the regulative cycle.

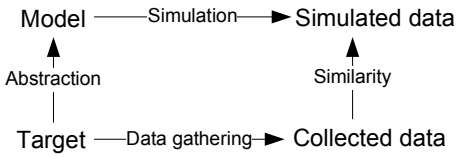


Figure 1.1: The logic of simulation as a method (Gilbert & Troitzsch, 1999).

First, the researcher creates a model of the target (the problem situation), but instead of using statistical equations, the modeller builds a computer program (instrument) that is able to simulate the target (e.g. a social or cognitive process). Next, the computer program generates an amount of data that can be compared to the collected data and thereby validates the model. The main difference between both approaches is that simulation models address the mechanisms that lie behind the processes that determine the final outcome, whereas statistical models "typically aim to explain correlations between variables measured at one single point in time" (Gilbert & Troitzsch, 1999, p. 17).

[9]The empirical cycle contains the following steps: (1) Observation—the exploration of phenomena, (2) Induction—design of an exploratory model, (3) Deduction—deduction of testable hypotheses, (4) Testing—confrontation of hypotheses with empirical data, and (5) Evaluation—rejection, revision or corroboration of the model. The regulative cycle contains the following steps: (1) Problem definition—perceived discrepancy between an actual and normative situation, (2) Diagnosis—formulation of the problem, (3) Plan—design of solutions, (4) Intervention—implementation of the solution proposed, and (5) Evaluation—test if the gap between actual and normative situation has narrowed (cf. Mietus, 1994; Homburg, 1999; Faber, 2006).

[10]Such an instrument can be applied in step 4-Testing-of the empirical cycle. In our case, the instrument serves as a tool that allows for generating data and comparing (testing) the simulation data with empirical data.

1.3.1 The design-based approach

Simulation models require a different approach than statistical models. In statistical models, the techniques applied for validating statistical models are well known by researchers in the field of social sciences and psychology and their corresponding tools are often supplied by software vendors (e.g. SPSS[11]) or open-source foundations (e.g. the R project[12]).

The techniques of computer simulation and more specifically the use of multi-agent simulations is not so widespread among researchers in the fields of psychology and social sciences. The use of programs to simulate outcomes of a model is known from the engineering disciplines, e.g. informatics, system engineering and artificial intelligence.

When creating statistical models, researchers apply off-the-shelf software that provides tools for analysing outcomes, whereas researchers that create simulation models develop their own programs (possible with help of existing tools) that require a different approach, i.e. it requires an engineering/design-based approach. The development of a program is similar to the engineering approach of constructing a building. First an architect creates a design (the design requirements), then an engineer creates the actual design, and finally, construction people build it. During the design process, from architect till construction worker, the necessary tests are carried out for verification and validation of the construction, resulting in iteration, redesign and a possible change of plans and design.

As a methodology to design and implement a Multi-Agent System, we have adopted the *design-based* approach (similar to the regulative cycle) (Beaudoin, 1994; Wright, Sloman, & Beaudoin, 1996), which specifies that:

1. The requirements of the system need to be specified;
2. Proposals of design need to satisfy those requirements;
3. Designs need to be implemented in a computer-simulation or in hardware;
4. The manner and the extent to which the design meets the requirements, and how the simulation embodies the design need to be analysed; and
5. The space of possible designs surrounding the proposed model should be studied and the design, which originates from an iterative process leading to the attainment of all its requirements, should be reconsidered.

The development of a model is described by Gilbert and Troitzsch (1999) as follows:

> One starts by identifying a 'puzzle', a question whose answer is not known and which it will be the aim of the research to resolve... This leads us to the [specific] *definition* of the target for modelling [stage 1]. [Next,] some *observations* [or collections of data] of

[11] http://www.spss.com
[12] http://www.r-project.org

> the target will be required in order to provide the parameters and initial conditions for our model. One can then make some *assumptions* and design the model [stage 2], probably in the form of a computer program [stage 3]. The simulation itself is performed by executing this program and the output of the simulation is recorded. [Next, w]e need to ensure that the model is correctly implemented and working as intended. This is *verification*—[a step in which the program is debugged—and]. . . *validation*[stage 4], ensuring that the behaviour of the model does correspond to the behaviour of the target. Finally, one needs to know how sensitive the model is to slight changes in the parameters and initial conditions: *sensitivity analysis*. (Gilbert & Troitzsch, 1999, pp. 17–18)

The development of a simulation model is not a strict linear process but rather a cyclical process whereby, after stage 4, stage 5 is entered and the development of the model often starts of again by adjusting the requirements or changing the computer model. This process is called the life cycle of a simulation study (Balci, 1998).

1.3.2 Life cycle of a simulation study

The life cycle of a simulation study exists out of several stages as pointed out in the previous section. This cycle is shown in figure 1.2 (cf. Bosman, 1977; Mitroff, Betz, Pondy, & Sagasti, 1974; Robinson, 2004; Balci, 1998).

The stages or phases in the model are comparable to the design stages discussed in the previous section. The *Real World* is the idea, situation and/or phenomena to be modelled; the stage in which the requirements are specified based on the real world. Then, the *Conceptual Model* is the mathematical/logical/verbal representation of the problem; proposals of the design are created that attempt to satisfy the requirements. The *Computer Model* (e.g. RBot, ACT-R or CLARION) is the implementation of the conceptual model as a program in a computer. The *Experimental Model* or models are specific implementations of the computer model that suits a particular study of the real problem, e.g. after the creation of a computer weather model, the experimental model of the weather in Groningen (Netherlands) is the specific application of the weather model to the situation in Groningen. The next stage is doing simulation runs and generating experimental data, and the interpretation, presentation and understanding of the data produced by the experimental model. The data can be compared to the real situation after which the model (and theory) can be redefined. The developmental cycle should not be interpreted as strictly sequential, i.e. the circle is iterative in nature and reverse transitions are expected (Balci, 1998).

Thus, besides the conceptual model, the following models are distinguished in figure 1.2: the experimental model—a model destined for a specific target and that only works specifically for that target—and the computer model—an (more) abstract model or toolkit that allows for the development of a range of specific experimental models that make use of the basic principles of the abstract

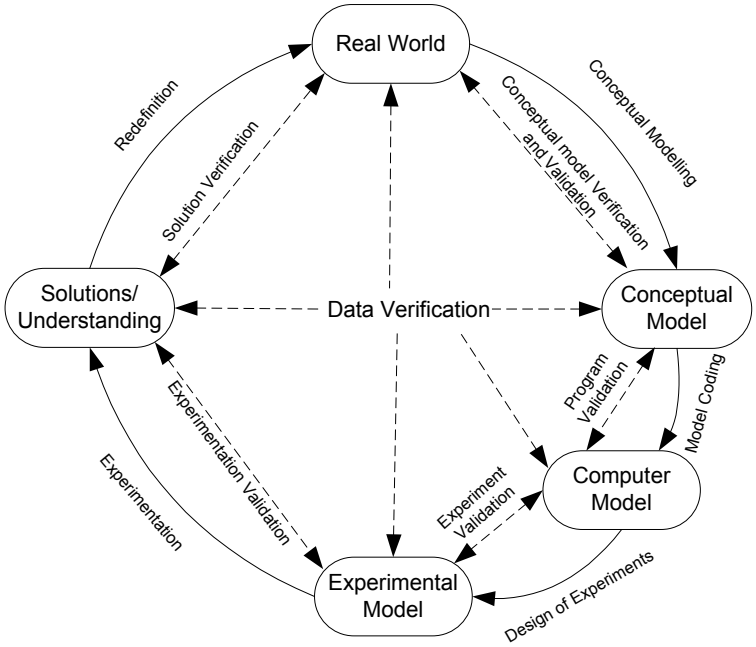


Figure 1.2: Life Cycle of a Simulation Study (adapted from Balci, 1998; Robinson, 2004).

model. In case of an abstract model (in our case RBot/MRS, see chapter 5), we follow a similar path in developing a model. However, the emphasis is not on a perfect fit with specific observed data, but more on the fit of mechanisms and phenomena that can possibly match with some real social phenomena.

Gilbert and Troitzsch (1999, p. 19) state that there is a continuum from detailed to abstract modelling and that abstract modelling[13] is research on 'artificial societies', i.e. simulation without reference to any specific 'real world' target. Or as stated by Epstein and Axtell (1996, p. 4):

> We view artificial societies as *laboratories*, where we attempt to 'grow' certain social structures in the computer—or *in silico*—the aim being to discover fundamental local or micro mechanisms that are sufficient to generate the macroscopic social structures and collective behaviours of interest.

In our demonstrative experiments (see chapter 6; experiment 2 and 3), we also create an artificial society and study social behaviour based on interaction between cognitive plausible actors (RBot), i.e. the actor's cognitive mechanisms are based on ACT-R, which is validated with help of empirical data. However,

[13] Abstract modelling in the sense that there is no concrete world to match the model with, but of course there are still concrete implementations of experimental models necessary to demonstrate the working of the abstract model.

the social phenomena that evolve from interactions between those actors are without reference to the real world.

The aim of our dissertation is to develop an abstract model or toolkit, whose purpose is threefold:

1. It can be applied for modelling and understanding of social and cognitive mechanisms that occur between actors and within actors, i.e. the simulation toolkit should allow for easy modelling (create experimental models) of interactive situations between actors delivering results that explain behaviour at the following levels: the social and interactive level between individuals, the level of the individual and the intra-individual (cognitive) level.

2. The second obvious step (and beyond the scope of this dissertation) is the modelling of specific situations that occur in reality, i.e. the gathering of data of interactions between organisations or people in the real world and comparing those with the simulation outcomes.

3. The third option is the application of the model in the virtual world of for instance e-business, gaming industry and many more situations. In these situations, the simulation program becomes an (pro) active agent that takes part in a virtual organisation or artificial society of interacting agents.

The design of our model aims to fulfil the first purpose and thereby supports other specific research as mentioned in point two and three, i.e. the design (RBot/MRS explained in chapter 5) is an abstract model or tool (purpose 1) that allows for the creation of many models that are suited to specific situations (purpose 2 and 3) to which those models should apply. The requirements of the abstract model are that they allow for the design of Multi-Agent Systems in order to explain interactive social behaviour between and based on *cognitive plausible actors*. The abstract model should give freedom for the creation of a range of diverse models—simple as well as complex models.

1.4 Research questions and objectives

Social (and organisational) behaviour is an outcome of the interactions between individuals. These interactions can lead to (temporarily) stable patterns of (organisational) behaviour. An *organisation* can be seen as a group of people that has habits of action aimed at cooperation and coordination of work. An organisation is not a physical, tangible object like an apple or a computer keyboard. Its observation, demarcation, and existence are dependent on the existence of human habits and human-produced signs; the organisation is a product of interactive social behaviour (Helmhout et al., 2004). An organisation can be traced back to representations (in the mind of actors) that structure the interaction among people, thereby demarcating a group of people who are members of an organisation (Van Heusden & Jorna, 2001).

As mentioned before, the aim of the dissertation is to design and implement a cognitive agent-based computational social simulation model to explain interactive (social) behaviour based on a selection of theories. This leads to the following research questions that are divided into (1) theoretical research questions and (2) design and implementation questions.

1 What are the aspects of actors that plausibly explain interactive (social) behaviour?

To answer this question, we need theories, models and requirements that address the individual as well as the interaction between individuals. The following three sub-questions will elaborate this:

1.1 What type of a model can explain interactive (social) behaviour?

Methodological individualism and social constructivism argue that social behaviour should be explained in terms of individuals, their properties and their interrelations in terms of these properties. Multi-Agent System as a model and methodology (see chapter 2) supports this view and can serve to explain interactive social behaviour; MAS is concerned with the study of behaviour, and the modelling of a collection of agents that interact with each other and their environment (Sycara, 1998).

1.2 What is required for an actor to exhibit (stable) social behaviour?

For an actor to exhibit social behaviour, it requires to create, interpret and exchange signs and meaning with other actors. Although theories of MAS address coordination, cooperation and negotiation issues, they focus only on social structures and overlook processes that describe how these structures emerge, stabilise and disappear. Whereas social constructivism and organisational semiotics (see chapter 3) consider that (stable) social behaviour requires (1) the creation of shared knowledge and the social construction of reality (cf. Berger & Luckmann, 1967), and (2) semiotic resources such as signs/symbols/social constructs that refer to social structures, institutions and habits of action.

1.3 What kind of an actor can plausibly handle signs, relations and social constructs?

An actor that handles signs, relations and social constructs needs to have a system that supports representations and mechanisms that manipulate these representations and exhibit intelligent action. More strongly, a general intelligent actor (system), whatever additional structures and processes it may have, will contain a physical symbol system (Newell & Simon, 1976; Newell, 1980). Therefore, the actor should be equipped with a cognitive architecture (see chapter 4). Such a cognitive plausible actor is able to generate and process symbols, create and 'understand' social constructs and build relations with other actors.

The theoretical research questions (1.1 until 1.3) have addressed the three main requirements necessary for the implementation of a cognitive agent-based social simulation tool:

- A simulated environment that provides a physical, communication and social environment for actors to live in.
- Social constructs, which are signs that serve as holders for sharing and reinforcing social norms, structures and institutions.
- A plausible cognitive agent that consists of a symbol system and cognitive/learning mechanisms that support the creation, manipulation, transfer of symbols/signs and is able to exhibit intelligent action.

Against this background, we can design and implement a multi-actor system. This task can be expressed in the form of the following research question.

2 How can (cognitive and social) plausible actors be implemented in a Multi-Agent System?

First, the cognitive actor, which is the main component of a MAS, is constructed. A production system is chosen from a selection of different approaches that are possible within AI (cf. Ferber, 1999, p. 125). It is the most elaborate choice that matches the necessary requirements for an actor to be a rational problem solver. A realistic computerised model of a cognitive plausible actor could be created with two cognitive architectures, either SOAR (Lehman et al., 2006; Newell, 1990) or ACT-R (Anderson & Lebiere, 1998). ACT-R was chosen here as the architecture to model the individual actor (see chapter 4 for a discussion). In its most recent release, ACT-R is an architecture of a single agent for which no multi-agent architecture version is yet available. Therefore, the ACT-R architecture is transformed and extended into a new architecture, named RBot. Together with a newly created MAS environment (Multi-RBot System or MRS), it is possible to incorporate multiple actors in a task environment.

In order to test the validity of a cognitive actor (RBot) as part of the new model (MRS), a comparison[14] with ACT-R is made in this dissertation. Consequently, the accompanying question is:

2.1 Is RBot comparable to ACT-R as a valid cognitive plausible actor?

After modelling the single actor and the platform on which to run multiple actors, a test is performed to see whether social constructs can emerge from the interaction of actors. The accompanying task environment is based on a well known experiment of *traffic laws* (Shoham & Tennenholtz, 1992a, 1995). In our task environment, actors face each other and need to pass each other to make progress. They have two choices. They can either drive left or drive right. Collision detection allows actors to react to each other's behaviour and, if necessary, to correct their behaviour.

Computational & Mathematical Organization Theory serves as an enlightening example of how organisational and social aspects (e.g. the traffic laws) can

[14]Multiple experiments were conducted to compare RBot with ACT-R. One of these experiments is demonstrated in this dissertation (see experiment 1, chapter 6).

be modelled with the help of MAS. CMOT is a multi-level system that is descriptive at the individual level—cognitive plausible actors as boundedly rational individuals (March & Simon, 1958)—and normative at the second level, i.e. the rules of the game such as defining different roles and authority structures. However, models of CMOT such as Radar-Soar (Carley et al., 1998) have the drawback that the macro-features of the model are pre-programmed (off-line) and do not emerge during interactions. The approach in this dissertation is to see whether behaviours, i.e. social constructs, can emerge from the interactions of agents, without the need to pre-program behaviour. An experiment is designed based on the proposed task environment to demonstrate the emergence of social constructs. The question that the experiment tries to answer is:

2.2 Is it possible that social constructs and organisational behaviour can emerge from social interaction?

In many situations, the interaction between agents is guided by social constructs that are formed as a habit between actors. The society is filled with rules (March, Schulz, & Zhou, 2000) that emerge from interactions, take on an implicit or explicit form and survive until they are no longer exercised. Coordination and cooperation is only possible when agents that interact with each other are aware of each other's social needs. In other words, the actor should have not only a representation of itself and the environment, but also that of (i.e. the needs of) the others; the society as a whole. The model of the cognitive agent is enhanced by adding a normative level of social constructs as representations to the actor rather than to the overall system. The question that the last demonstrative experiment tries to answer is the following:

2.3 Is a social construct a coordination mechanism that can influence the behaviour of interacting related actors towards a certain desired behaviour?

The objective of the design and implementation questions (2.1 until 2.3) is to address the mechanisms inside the actor that are influenced by interactions with other actors and thereby socially constructed. The underlying assumption is that the actor is socially connected to others on a continual basis. The second objective is that these questions are simultaneously design questions. First, the requirements are drawn up. Second, the design or model is constructed. This is followed by an implementation of the system, and fourth, demonstrative experiments show its working. The design questions guide the construction of a model that attempts to bridge the gap between cognitive and social sciences (cf. Sun, 2006a).

1.5 Dissertation outline

The dissertation is structured in the following way (see figure 1.3). The introduction, i.e. the current chapter, explains the theoretical disciplines that are used and how these disciplines are applied to the research, discuss the methodology, followed by the research questions, objectives and the dissertation outline.

Chapter 2 commences with a general introduction to Multi-Agent Systems. The chapter explains the notion of agency regarding different types of agents and how these agents interact with each other in an environment. It gives the reader an overview of the literature in the area of MAS.

Chapter 3 introduces the social actor and explains several theories including the social constructivist theory and organisational semiotics that bridge the gap between social (behavioural) and cognitive science. The theory describes how actors can be embedded in organisations and society. Secondly, based on these theories, a model of the social actor is presented, i.e. a model that is reactive, pro-active and socially and physically situated in the environment.

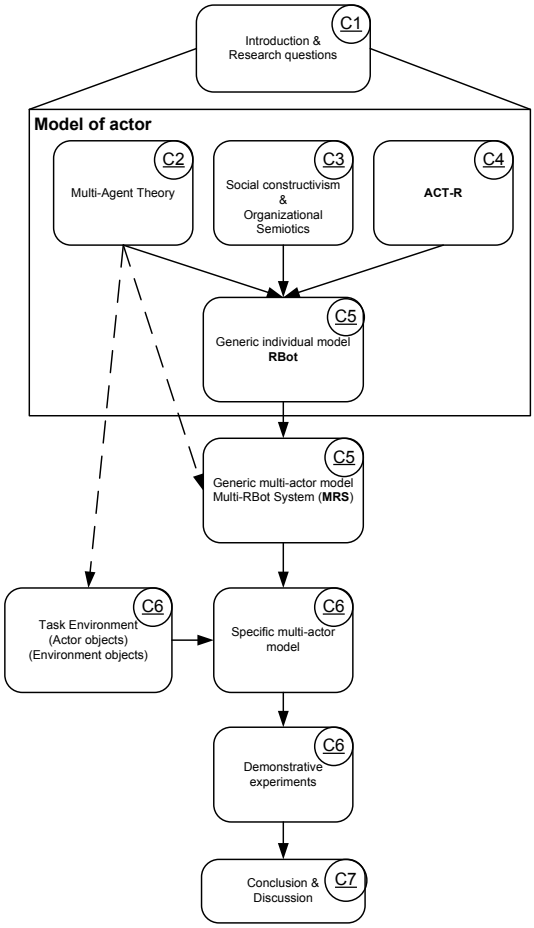


Figure 1.3: Structure of the dissertation.

In chapter 4, the selection of a cognitive architecture is discussed. The cognitive architecture ACT-R is selected as the architecture for constructing a computer model (RBot) of a cognitive plausible actor. ACT-R is discussed in detail, because it is the main contributor to the new RBot model. The subsumption architecture (Brooks, 1986) is drawn upon as a supporting theory to implement social behaviour at a normative level.

In chapter 5, the design chapter, we will discuss (1) the architecture of the social and cognitive actor (RBot), and (2) the multi-agent environment (MRS). First, the actor is modelled in a bottom-up approach starting with the memory model, then the controller and handlers, followed by the implementation of social constructs. Second, the server (as part of MRS) is modelled in which a controller regulates (1) registration of actors, (2) the time synchronisation between actors happens, (3) the communication between actors with the help of a message-based architecture is regulated, and (4) the physical, communication and social environment is maintained.

In chapter 6, the three experiments are discussed. The first experiment focuses on the functioning of the single actor. The experiment shows that RBot processes 'symbols in the mind' similar to the way ACT-R treats those symbols. This is done by demonstrating a multi-column addition problem in which RBot

behaves similar to ACT-R in solving this problem, i.e. with help of a goal stack, a set of procedures and a declarative memory.

The other two experiments demonstrate the working of RBot as well as its benefits in explaining the emergence and influence of social constructs as a coordination mechanism. The first one of these two experiments is a traffic situation in which two actors confront and try to pass each other. Both actors have a similar set of procedures. However, they have a choice of selecting either the left or the right side to pass the oncoming driver/actor. The interaction between the actors and the individual experience of success and failure in passing the other emerge into a stable behaviour over time. The assumption is that both actors/drivers build up a preference, or come to a tacit mutual agreement that is in favour of both actors. The other, and the last, experiment is an extension of the previous traffic situation. However, in this experiment, one of the actors acts as an authority (a policeman assigned by society) to communicate laws or social constructs to other actors. Compared to the previous experiment, the authoritative actor in the last experiment is equipped with a mechanism that supports it in conveying the 'rules of the game' to influence the behaviour of others when rules are not abided by.

Finally, chapter 7 initially presents the contributions of the research and it draws general conclusions bearing in mind the research questions addressed in this chapter. Second, it advocates the importance of cognitive actors in increasing plausibility in explaining social behaviour and the empirical testing of cognitive models to enrich social simulations. Third, new questions are raised and last, the chapter highlights avenues for further research.

CHAPTER 2

Multi-Agent Systems

THE first chapter has discussed the issue of how cognitive science can contribute to social sciences in order to explain interactive social behaviour. In chapter 5, we will develop a new cognitive agent-based computational social simulation model (RBot) in an attempt to satisfy the need for new theories and models. In this chapter, we will discuss Multi-Agent Systems (MAS) as one of the theories for the development of such a new model.

Multi-Agent Systems (MAS) is the name for a new approach of designing, analysing and implementing complex adaptive software systems (Jennings, Sycara, & Wooldridge, 1998). This approach has emerged out of a history of theory and applications in the area of (social) simulation models.

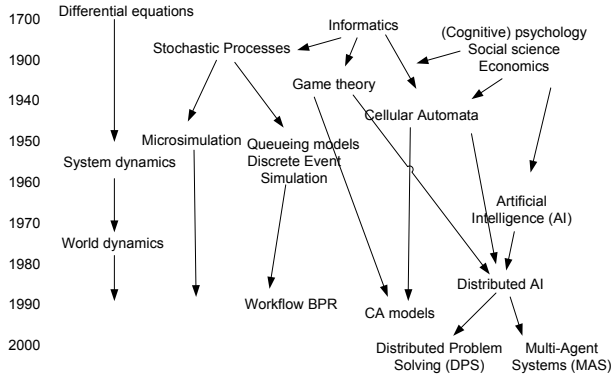


Figure 2.1: History of contemporary approaches of (social) simulation theories and techniques (adapted from Troitzsch, 1997; Gilbert & Troitzsch, 1999).

Figure 2.1 shows a selection of simulation theories and techniques that have been developed over the years and many of them have been applied in the field of MAS and social simulations. The early development of simulation techniques at the left side of the figure are primarily based on mathematics. Moving to the right, there are simulations and theories that exist because of the introduction of the computer and at the bottom right, we can see the influence of Internet on the upcoming popularity of Distributed AI and Multi-Agent Systems.

Mathematical theories, e.g. differential equations and stochastic processes, require specific skills for describing deterministic and stochastic processes in an attempt to predict or describe phenomena such as the dynamics of populations (Volterra, 1926). Game theory, discussed later on in this chapter, is a form of applied mathematics to understand economic behaviour based on interactions or games between rational agents who try to maximise their profits. Game theory still has great influence in describing outcome strategies in simulations.

In the 1960s, the first computer simulation models were developed with help of queueing models, discrete-event simulations (Birthwistle, Dahl, Myhrhaug, & Nygard, 1973) and system dynamics (Forrester, 1991). Around the same time, micro-simulation or micro-analytical simulation models (MSMs) emerged as a simulation technique.

> [MSMs have] been used to predict individual and group effects of aggregate political measures which often apply differently to different persons... [They] consist of at least two levels: the level of individuals or households (or the level of enterprises) and the aggregate level (for example, the population or national economy level). (Gilbert & Troitzsch, 1999, pp. 53–55)

Another approach that emerged was cellular automata whose behaviour emerges from properties of local interactions. The basic model of a cellular automata is a two-dimensional grid of cells—the structure of squared paper—in which a cell reacts only with its neighbouring cells. The simulation uses a time-step generator and at every time-step, every cell reacts to its neighbours according to a set of rules. In this way a stepwise reaction will flow through the grid and influences the overall behaviour of the system.

> To summarise, cellular automata model a world in which space is represented as a uniform grid, time advances by steps, and the 'laws' of the world are represented by a uniform set of rules which compute each cell's state from its own previous state and those of its close neighbours. (Gilbert & Troitzsch, 1999, p. 122)

In the 1980s, artificial intelligence, a separate field concerned with intelligence of the individual, became interested in the interaction and distribution of intelligence, known as Distributed Artificial Intelligence (DAI). Ferber (1999, p. 24) gives a short description about the history of the roots of DAI that today has resulted into Multi-Agent Systems. DAI started around the eighties with the model of the blackboard system (Erman, Hayes-Roth, Lesser, & Reddy, 1980); a model consisting of Knowledge Sources (KSs) organised in a star-topology with

in its core the 'blackboard'—a place where the KSs can share their knowledge—and which is managed by a separate control device that has the task to prevent and coordinate conflicts of access between these KSs. Around the same time another type of control system (cf. Lenat & Brown, 1984; Lenat, 1975) came up that solved problems with help of a community of specialists. Hewitt (1977) described control structures in which he did not consider processes as a sequence of choices, but "he tended to think in terms of distributed systems considering control structures as patterns of message passing between active entities called *actors*" (Ferber, 1999, p. 25) [1].

The roots of DAI are the foundations of today's two approaches in DAI: Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS) (Bond & Gasser, 1988). DPS takes a pure *engineering* approach to distributed systems, i.e. it is concerned with "how to build functioning, automated, coordinated problem solvers for specific applications (Bond & Gasser, 1988, p. 4)" (as cited by Van den Broek, 2001, p. 21). The approach taken by DPS is a top-down approach in which tasks are decomposed into smaller tasks appointed to specialists.

> A DPS is... a top-down designed system, since agents are designed to conform to the problem-solving requirements specified at the top. Within this top-down task decomposition approach, the individual components are considered to be of *secondary importance* to the need of the overall system. The agents themselves have *limited autonomy* because their role in solving the overall problem is usually designed-in, with coordination rules included. (emphasis added: Van den Broek, 2001, p. 22)

On the other hand, a MAS is constructed taking the bottom-up approach. The agent itself is now the centre of attention and not the system as a whole. In a MAS, the agent has more autonomy, and control is delegated to the agent, i.e. the overall outcome of solving the problem is not determined by a strictly top-down organised control system, as in the case of DPS, but by (social) interactions between independent and rational agents making decisions that satisfy their own (be it social) goals.

[1] There is not a clear definition about or difference between the terms 'actor' and 'agent'. For instance, Hewitt gives the following description of the actor: "The actor metaphor for problem solving is a large human scientific society: each actor is a scientist. Each has her own duties, specialties and contracts. Control is decentralized among actors. Communication is highly stylized and formal using messages that are sent to individual actors" (Hewitt, 1977, p. 350). Apart from the description, he gives the following remark: *"The reader should keep in mind that within the actor model of computation there is no way to decompose an actor into its parts. An actor is defined by its behavior; not by its physical representation"* (Hewitt, 1977, p. 327).
In this dissertation, we will use the metaphor actor in a different way; as an entity that can be decomposed into (cognitive) elements that have their own behaviour. Besides that, we will also adopt the term 'agent', because of its wide adoption in the field of distributed systems. In section 2.1, we will explain the notion of agency that describes the term agent in terms of the weak and the strong agent. Although the distinction between 'actor' and 'agent' is not that sharp, and can often be synonymous, in this dissertation we try to distinguish both by referring to the term 'actor' as a being that is more autonomous and is a more general concept than the 'agent'. The agent often refers to a software entity that serves the person who designed the agent, e.g. an Internet search-agent, insurance agent, etc.

In this dissertation we adopt MAS as theory and methodology for simulating organisations for the following reason:

> ...[M]ultiagent systems applied within the natural systems approach[2] provide the required methodological position in simulating organisations. The multiagent system approach exploits the full potential of the DAI paradigm because of its emphasis on the emergence of a system level and a strong notion of [agency[3]] resulting in higher autonomy levels in agents. In this way, the agents become more socio-realistic in that they might compete and disagree with other agents, and generally act in their own best interest. The result is that coordination among the agents is necessary in order to achieve coherent collective behavior. Hence, multiagent models perceived in this way are bottom-up models of coordinated problem solving. (Van den Broek, 2001, p. 24)

In other words, MAS can supply us with the means to study what the aspects of actors are that plausibly explain interactive (social) behaviour (see chapter 1; research question 1). Hence, MAS is concerned with the behaviour of a collection of distributed autonomous (heterogeneous) agents aiming at solving a given problem. The characteristics of a generic MAS are (Jennings et al., 1998):

- Each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint;
- There is no global system control;
- Data is decentralised; and
- Computation is asynchronous

Inferred from these characteristics, the main component studied in Multi-Agent Systems is the autonomous *agent* and its associated behaviour in an environment.

The approach taken in this chapter is to explain MAS starting with the individual agent and subsequently going towards a system of (cognitive) social agents that considers relations, coordination and interaction between agents. The purpose of the chapter is not to provide a detailed review of the field of MAS, but to accustom the reader with the paradigm MAS[4].

This chapter is structured as follows. First, in section 2.1, the notion of agency is discussed to clarify the term agent. The term agent is applied in many research fields, e.g. biology, economics, cognitive science, social science and so on, and therefore has become somewhat blurred. In section 2.2, we approach the agent as a human-like entity in which components, e.g. perception,

[2] "A natural systems approach... stud[ies] the strategies and representations that people use to coordinate their activities, in much the same way the cognitive scientists investigate individual cognition in people" (Van den Broek, 2001, p. 23).

[3] The strong notion of agency will be explained in the next section.

[4] See for additional information Bond and Gasser (1988), Durfee, Lesser, and Corkill (1992), Jennings et al. (1998), Sawyer (2003), Sycara (1998) or Wooldridge (2002).

cognition, action and so on, form the basis for constructing an agent architecture. Section 2.3 describes two agent typologies: the cognitive agent commonly used in AI and the social agent. Whereas the cognitive approach focuses on the internal working of the agent, the social approach studies how (socially) well an agent is embedded in its social-cultural environment. The environment plays an important role in the design of an agent; it, in part, defines how many degrees of freedom an agent possibly can have. In section 2.4, the distinction between a physical, communication, social and task environment is described. An environment allows agents to interact with other agents and at the same time, scarcity of space, time and objects in an environment can create conflicts between agents. In section 2.5, several types of coordination mechanisms are discussed that enable conflicting agent to negotiate and cooperate. Many situations in our daily life are solved with the help of coordination mechanisms. For instance, traffic lights function because we understand the meaning of the coordination mechanism. Such understanding solves problems that otherwise would occur when several cars want to occupy the same space (crossroad) at the same time. Section 2.6 discusses the applications that are implemented in the field of MAS. They vary from simple desktop applications for consumers, towards complicated distributed real-time systems in the industry. Applications of MAS are—especially on the Internet—widespread and have proven themselves a worthy competitor to alternative systems, i.e. due to the Internet, MAS applications are more wanted than ever before and can be the answer to many of today's coordination problems. Finally, section 2.7 closes the chapter with a discussion about what kind of agent and theory is important for our research, i.e. the type of agent will be selected from a variation of agents that differ in the way they are constructed, from simple to very complex. The complexity of such a construction depends on the philosophical ideas behind the design of the agent and its environment, and the amount of complexity we need in order to answer the research questions.

2.1 Agency

Understanding the field of MAS starts with the understanding of the agent as the main component of the system. In different fields, researchers apply the term agent differently and even within MAS, there are different understandings of the term agent, i.e. the context of application is necessary to understand the meaning of the term agent. Many definitions and classifications of agents have been proposed (Franklin & Graesser, 1996). In this dissertation we first introduce a general definition of an agent: "An agent is a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives" (Wooldridge, 2002, p. 15). Starting from this point, Wooldridge and Jennings (1995) distinguish two notions of the term agent: the first is the weak notion and the second the strong notion of agency.

2.1.1 Weak notion of agency

The weak notion is a general way to describe the term agent as a hardware or software-based computer system with the following properties:

Autonomy: agents operate without direct intervention of humans or others, and have some kind of control over their actions and internal state (Castelfranchi, 1995).

Social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language (Genesereth & Ketchpel, 1994).

Reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it.

Pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.

The weak notion is applied in situations where behaviours, in contrast to the strong notion, are described at a high level of abstraction.

2.1.2 Strong notion of agency

In the field of AI, the notion of agency is stronger and apart from the characteristics of the weak notion, agents have additional properties, such as mental attitudes—knowledge, beliefs and so on—or representations (Jorna, 2002), i.e. an agent is "...an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments. These components are defined in a precise fashion,..." (Shoham, 1993, p. 52). Strong agents are often defined in more detail and can be described at the intentional or the functional level of description. Dennett (1987, 1978) introduced a distinction in various levels of description. He discerned a physical, a functional and an intentional level (or stance) of description.

> The physical stance explains behavior in terms of physical properties of the states and the behavior of the system under concern. For its proper functioning the human organism requires a complex interaction between its parts and with the external world... The second level of explanation takes the point of view of the functional design of a system. The behavior of a system is conceived of as the result of the interaction between a number of functional components or processes. The physical structure (architecture) of the system is not explicitly taken into account, although it may impose constraints on the behavior of the system. In the third place Dennett distinguishes the intentional stance. Complex behavior that is adapted to the prevailing circumstances, according to some criterion of optimality is said to be rational or intelligent. A behaving system to which we can successfully attribute rationality or intelligence qualifies as an intentional system. (Gazendam & Jorna, 2002, pp. 12–13)

The intentional level and functional level are considered as appropriate for applications in agent theory to ascribe mental attitudes and cognitive mechanisms. Two types of strong agents can be distinguished. The first type is the reasoning agent that is defined at the intentional level and whose implementation is based on logics and the practical reasoning theory (Bratman, 1987). The other agent is an agent with a cognitive architecture, or a symbol system and is defined at both, the intentional and the functional level. The former type of agent is used in primarily describing behaviour of agents, and the latter is applied when more details have to be known about the internal functioning (implicit behaviour) of the agent.

In order to better describe the strong agent, Newell (1982) developed a systematic description of an agent's behaviours known as the computer system level paradigm. The computer system level paradigm describes five levels of description, the device, the circuit, the register transfer, the symbol and the knowledge level. We are mainly interested in the knowledge level and the symbol level.

Newell applied the knowledge level to describe problems and the symbol level to implement and solve the problems. At the knowledge level, a problem can be described and solved by determining the goals of an agent and the actions that have to be performed to achieve these goals (Kalenka, 2001). At the symbol level, a physical symbol system (Newell, 1980) is applied for implementation and is necessary for exhibiting intelligent action.

The five levels of the computer system level paradigm can be connected to different timescales or bands (Newell, 1990). Whereas neurons may fire in 1 ms, and cognitive reasoning takes 100ms till 15 seconds, higher rational actions may take minutes or hours. Organisational and social behaviour can take up minutes, days, months, or even more than years.

Besides that, Newell (1982) introduced the behavioural law, the Principle of Rationality, which is applicable to the Knowledge Level: "If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action." (p. 102). However, Newell (1982)'s behavioural law only takes into account the individual and its desires, beliefs and goals.

As an addition for taking into account the society, Jennings and Campos (1997) defined a new system level above the knowledge level—the social level—in which the whole MAS or society is represented and consists of socially responsible agents. They define agents as being autonomous and balancing their individual needs with those of the overall system (Kalenka, 2001). According to Jennings and Campos (1997), the shortcoming of the behavioural law, i.e. the Principle of Rationality, is that it has meaning for a single agent and cannot be applied to a system of multiple agents. Therefore, Jennings and Campos (1997) extended the definition with a new principle that includes society; the Principle of Social Rationality: "If a member of a responsible society can perform an action whose joint benefit is greater than its joint loss, then it may select that action." (p. 16). This very strong notion not only assumes that the individual is self-interested, but also socially responsible for its actions.

In many Multi-Agent Systems agents are relatively weak, e.g. the well known predator-prey simulations based on the mathematical model of Lotka

and Volterra (1926). MAS agents are often constructed at a level that shows no internal mechanisms and therefore can only describe behaviour based at that (the intentional) level.

On the other hand, cognitive science tries to explore the functional level—see SOAR (Newell, 1990) and ACT-R (Anderson & Lebiere, 1998)—and even tries to touch the physical stance (connectionism). However, the weak spot of cognitive science (cognitive psychology) is its focus on the detailed individual organisation of the agent.

Therefore, the combination of cognitive science and MAS can create new opportunities for explaining cognitive and social behaviour varying from the individual level (functional and rational) to the social level. The strong notion of agency is necessary to explain behaviours at the intentional level (with help of the functional level) in a cognitive plausible way, i.e. an agent should be designed with the right grain size in order to have the predictive and explanatory power necessary to explain the phenomena studied. Apart from describing individual mental attitudes, social (and possible cultural) attitudes (Jennings & Campos, 1997) should be cognitively embedded as well in order to explain why an agent sometimes is obliged or prohibited to take actions that incur personal loss, but societal benefit.

2.2 Agent components and architecture

The notion of agency implies that there are different types of agents or architectures that describe how an agent (architecture) could be designed. The most common method used to discern agents is to define components and requirements that assign functions. The construction of an agent architecture based on these components determines the type of agent (cf. Conte & Castelfranchi, 1995b; Russell & Norvig, 2003; Van den Broek, 2001; Verhagen, 2000; Wooldridge, 2002).

Three main (cognitive) components for constructing the mind of the agent are often distinguished: perception, action and cognition. The peripheral components, i.e. perception and action, and the interaction between those components are discussed first.

2.2.1 Perception and attention

Perception is one of the components necessary to receive input signals from an environment, especially when the agent depends on the environment and the environment is constantly changing. Ferber (1999) distinguishes two types of perception, *passive* and *active* perception. With passive perception, the input signals are registered and classified accordingly. Active perception is a complex mechanism that combines and compares sensory data with data based on expectations, goals and experiences that are stored in the cognitive system. Active perception also can give specific instructions, based on classification and affordance, for tracking changes in the environment, e.g. which objects are able to move. In a complex environment, the agent should have some mechanisms that filter out information making it possible for the agent to give attention only to

senses that can be assigned as relevant input. Such an attention mechanism can be hard-coded in the design of an agent, by for instance giving it limitations for frequencies for hearing, or a certain range of colours in the spectrum of light.

2.2.2 Action

Actions are elements or means that enable the agent to change states. States can be situated and designed in the agent self, the environment or perceived and attributed to other agents. The number of actions is restricted by the agents limited number of known actions—cognitive limitation—and the allowance for these actions by the environment—physical limitation; e.g. the agent decides to throw a stone, but no stone is available. Van den Broek (2001) acknowledges this and states that the environment provides the agent a context in which actions have meaning and can be performed or not. Action, therefore, is a relative notion—known as *situated action* or *situated cognition*.

2.2.3 Interaction between perception and action

Susan Hurley (2001) discusses several approaches to the interaction of perception and action: the traditional view, the behavioural view, the ecological view, and her own two-level interdependence view. She distinguishes the subconscious sub-personal level from the conscious personal level. At the sub-personal level, there are sensory inputs and motor outputs. At the personal level, there are perceptual content (perceptions) and intentional content (intentions).

She explains that in the *traditional view* of the relation between perception and action, there is a vertical separation between perception, cognition and action. The mind passively receives sensory input from its environment, structures that input into perceptual content, processes this content in the cognitive system creating intentional content, and sends this intentional content to the motor system giving motor output. This is more or less a *linear* flow from environment through sensory system, cognition, and motor system and finally out to the environment again. Furthermore, sensory system and motor system are independent units that can have effects on each other through causal chains: the sensory system has effect on the motor system via the link perceptual units—cognitive processing—intentional units, while the motor system causes the position of the body or the environment to change, an effect that can be picked up by the perceptual system. One could say that this is an *instrumental* connection: "perception is a means to action and action is a means to perception" (Hurley, 2001, p. 12).

This traditional view on perception and action has received critique from two directions. First, instead of assuming a linear view, perception and action are seen to be more closely linked by dynamic feedback loops. Secondly, instead of a mere instrumental connection between perception and action, dynamical system approaches to the mind aim to show how cognition emerges as a function of an underlying dynamical system in which action and perception are constitutive parts. Hurley sees *behaviourism* as an approach that accepts the

linear view and rejects the instrumental view. It sees action and perception as constitutive parts of one functional unit.

Hurley places the *ecological approach* to perception and action as one that rejects the linear view but accepts the instrumental view. In this respect, it could be said that she does not understand Gibson's theory and wrongly places him in the corner of advocates of an instrumental link between perception and action:

> To place ecological views of the relations between perception and action into a box designating those relations as merely instrumental belies a fundamental misconstrual of Gibsonian psychology ...Most importantly, however, it is clear that ecological psychologists do not regard the relation between perception and action as merely instrumental: the very things they propose animals perceive are *affordances*, or opportunities for action. Affordances are neither features of the external world nor internal representations; they are neither objects nor mental phenomena. Instead, they are the relational properties between the available resources in an environment and an organism's abilities to utilize those resources. Therefore, if it is affordances that an organism perceives, as Gibsonians claim, then perception inherently involves action because the [...] very kinds of things perceived are relative to the actions an organism can perform. We stress that this is, moreover, a constitutive (i.e. non-instrumental) link between perception and action: an organism's perceptions are in part constituted by its behavioral repertoire. (Chemero & Cordeiro, 2006)

In contrast to the traditional view, Hurley explains her two-level interdependence view. She states that the relationship between perception and action is not linear, but consists out of numerous dynamic feedback loops. Furthermore, she argues that there is not a mere instrumental link between perception and action, but that perception and action are *constitutively linked*, that is, they are inseparable parts of a larger dynamic system: "The contents of perceptions and intentions are each constituted by processes involving both inputs and outputs. Both are functions (albeit different ones) of the same underlying relations between inputs and outputs" (Frankish, 2006).

Change in intentions can lead to a change in perceptions. Hurley argues that, therefore, there must be a constitutive link between perception and action and not a mere instrumental link. So, perceptual content is constitutively linked to sensory input processes and motor output processes, as intentional content is. Perceptual content and intentional content are thus also constitutively linked as parts of a larger dynamical system. A picture emerges of a sub-personal level and a personal level that are interdependent parts of a larger dynamical system: the *two-level interdependence view* (Hurley, 2001).

Constitutive links also exists between processes of sensory input and processes of motor output, leading to horizontally layered modules. These modules are directly coupled to the actor's environment:

> One way to think of these is in terms of layer upon layer of

> content-specific networks. Each layer or horizontal module is dynamic, extending from input through output and back to input in various feedback loops. Layers are dedicated to particular kinds of tasks. One network, for example, may govern spatial perception and the orientation of action (the so-called 'where' system). Another may govern food recognition and acquisition-type behavior (part of the so-called 'what' system). Another may govern predator recognition and fleeing-type behavior (another part of the 'what' system). Another may govern some of the variety of imitative responses to the observed behavior of others, and so on. Evolution and/or development can be seen as selecting for each layer. Each subpersonal layer is a complete input-output-input loop, essentially continuous and dynamic, involving external as well as internal feedback. Thus, not only are sensory and motor processes coupled, but the neural network is directly coupled to the creature's environment; horizontal modules are essentially 'situated'. (Hurley, 2001, p. 6)

With respect to this coupling of organism and environment, Hurley can be seen as an ultra-externalist by suggesting that perception content and intention content are constitutively linked to subpersonal processes that extend into the environment. (Frankish, 2006).

A critique on Hurley (2001)'s work states that she uses dynamical systems theory in a wrong way:

> Drawing on this dynamical cognitive science, Hurley says that a sub-person is a 'dynamic singularity'. These ideas are faulty because if sub-persons are well understood in terms of dynamical systems theory, then the ideas of input and output ... don't apply to them. Dynamic singularities include more than just the organism. There are often loops that go out into the external environment (as well as internal loops). These loops are part of the (sub-personal) dynamic singularity. But if the singularity is the whole dynamic system, the traditional ideas of inputs and outputs don't make sense. Inputs from what? Outputs to what? (Chemero & Cordeiro, 2006)

Hurley (2001) has made an interesting contribution with her two-level interdependence view. Especially the description of horizontally layered modules that consist of sensory input processes and motor output processes, that are based on dynamic feedback loops and that extend into the environment are interesting. These horizontally layered modules are constitutively linked to perceptual content and intentional content. We see that Hurley (2001)'s theory is more compatible with Gibson's theory and organisational semiotics than she thinks: her horizontally layered modules could correspond to the Gibson's physical affordances, while larger modules functionally dependent on these physical affordances could be the social constructs of organisational semiotics (see chapter 3). Furthermore, the extension of the horizontally layered modules into the environment can be seen as compatible with the theory of the semiotic Umwelt in semiotics (see chapter 3).

Although interaction between perception and action is interesting enough on its own, the focus of this dissertation is not on the complexity of perception or action but more on the social situatedness of the cognition. In this dissertation, a one-way circular system such as the traditional view can be applied for simplicity, i.e. complex perception-action systems go beyond the aim of this dissertation.

2.2.4 Cognition

Perception and action are enough when a system only has to control and adapt to the current situation. However, when an agent has the desire to improve over time with help of experience, then the agent requires intelligence or a cognitive system and a memory of representations and experiences from the past in order to be able to learn from previous actions. For now, we will postpone a detailed discussion about cognition and come back to it in chapter 4. In this section, we will describe the functional components comprising a cognitive system that requires at least memory, learning and goal-directed behaviour.

2.2.4.1 Memory

Assuming that agents have feedback loops for comparing actions with past actions and perceptions with past perceptions, then the agent should be able to maintain state or have a memory with experiences of past states. For instance, many economic agents that work with utility functions have a very short-term memory, only containing the outcome of the previous state, whereas deliberative agents have an underlying memory structure and management in which inferences of a longer period of time can be stored. Hence, a deliberative agent can store experiences and facts about the world around himself and with help of inferences and knowledge about "how the world works", a world-model can be constructed (Russell & Norvig, 2003).

Cognitive architectures, such as SOAR (Lehman et al., 2006) and ACT-R (Anderson & Lebiere, 1998) support memory models of long-term memory (LTM) and short-term memory (STM) (cf. Atkinson & Shiffrin, 1968). A taxonomy of memory types (see figure 2.2) distinguishes different types of long and short term memories (Anderson, 1976; Tulving, 1985; Lehman et al., 2006; Miyashita, 2004; Wilson, 1975)

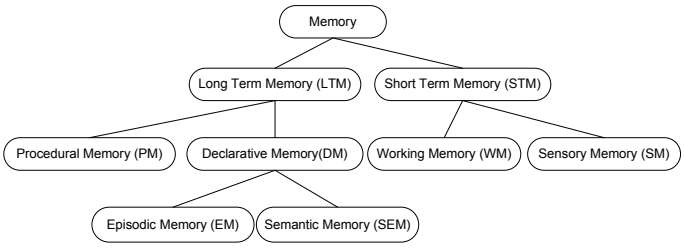


Figure 2.2: Taxonomy of memory types.

Long term memory (LTM), compared to STM, is able to store knowledge for a long time and in a cognitive architecture this memory can be subdivided into procedural memory (PM)—sometimes referred to as a rule-base—and declarative memory (STM), the storage of beliefs or facts. Procedural memory contains procedures and skills about how and when to act, whereas declarative memory contains facts and inferences[5] about the world. In declarative memory, a distinction can be made between semantic (SEM) and episodic memory (EM). Semantic memory is used to store meanings, understandings and factual knowledge about the world. The episodic memory (Tulving, 1972) is used for specific episodes with a reference to an event that happened at a certain time in the past and is often associated with emotions that strengthen its presence in memory.

Short term memory can be divided into sensory (SM) and working memory (WM). Sensory memory (Sperling, 1960) is a very short lived memory in which traces of input are stored for a while. Sensory memory can be detected in practise when looking at a fast turning wheel with spokes; the wheel seems to turn anti-clockwise because there are still past traces in memory.

Working memory (Baddely, 1986) is a storage unit for temporary information storage and manipulation. The capacity is limited (Miller, 1956), e.g. think about the use of paper when trying to calculate a complex equation.

In this dissertation, the procedural memory, the declarative memory and the working memory get the most attention. A further explanation of all other memory aspects is beyond the scope of this dissertation.

In current cognitive architectures like SOAR and ACT-R, the focus is on procedural, declarative (semantic) and working memory. In both architectures, the working memory functions as a placeholder for knowledge that is relevant for the current situation. Knowledge inside working memory is directly available, while knowledge in long-term memory first needs to be retrieved in order to be used. SOAR and ACT-R both describe procedural memory as the place where skills and rules (if $\rightarrow$ then) are stored that enable the actor to act when certain conditions take place. The function of the declarative memory is to supply missing information that a procedure needs to solve the problem. Scientists in the field of memory research, are especially interested in the way people recognise, recall and re-learn certain items in memory (Anderson & Lebiere, 1998). ACT-R allows to model these phenomena with help of sub-symbolic[6] activation in order to make memory degrade over time and simulate how people forget and learn, and how associative memory can increase latency times when more elements are added to memory (cf. the fan effect: Anderson, 1974). The properties of memory have an effect on the performance of remembering and learning.

2.2.4.2 Learning

Closely connected to theories of memory are theories of learning. Remembering, recognition and recall are processes that are inherent to processes of learning. Memory alone supports in collecting and maintaining knowledge, however there is a need to classify over time, based on criteria, what experiences were

[5]For example: All men are mortal, X is a man $\rightarrow$ X is mortal.

[6](cf. Smolensky, 1988)

successful or not. Learning defines these criteria that can be based on, for example, external or on internal goals, drives or beliefs, often with the help of utility functions. Following, Rumelhart and Norman (1978), three kinds of learning[7] can be distinguished:

> (a) *accretion*, or the encoding of new information in terms of existing schemata[8]; (b) *restructuring or schema creation*, or the process whereby new schemata are created; and (c) *tuning or schema evolution*, or the slow modification and refinement of a schema as a result of using it in different situations. (Shuell, 1986, p. 421)

Accretion and restructuring (also called symbolic learning) involve the creation of new knowledge (or symbols). With accretion, new knowledge is added to memory without changing the structure of memory. With restructuring, inferences or structures in memory are changed without acquiring new knowledge from the outside world. However, new knowledge can be generated or inferred from prior knowledge already present in the mind. Tuning of information is the (evolutionary) change of generalising or constraining the applicability of knowledge.

In a similar way, machine learning also offers different types of learning, e.g. supervised, unsupervised, and reinforcement learning (Russell & Norvig, 2003). Supervised learning concerns the construction of a function (induction) with the help of training examples delivered by the outside world. For instance, the supervised learner is able to classify objects that were previously unknown to the learner and the learner's reference is based on training examples delivered to him by the outside world. Unsupervised learning is the learning of associations or classifying about the outside world without having feedback from the outside world. For example, in machine learning, the Hebbian learning rule (Hebb, 1949) states that the connection between two neurons is strengthened if they fire at the same time.

While supervised learning assumes that there is a clear feedback teacher signal, reinforcement learning is learning what to do, i.e. how to map situations to action, in order to maximise a numerical reward signal. Reinforcement learning takes into account the trade-off between *exploration* and *exploitation*:

> A central concern of studies of adaptive processes is the relation between the exploration of new possibilities and the exploitation of old certainties (Schumpeter, 1934; Holland, 1975; Kuran, 1988)...Adaptive systems that engage in exploration to the exclusion of exploitation are likely to find that they suffer the costs of experimentation without gaining many of its benefits. They exhibit too many undeveloped new ideas and too little distinctive competence. Conversely, systems that engage in exploitation to the exclusion of

[7] These three kinds of learning are similar to the concepts 'assimilation', 'accommodation' and 'equilibration' of Piaget (1985).

[8] Schema(ta) represent knowledge of all kinds from simple to complex.

> exploration are likely to find themselves trapped in suboptimal stable equilibria. As a result, maintaining an appropriate balance between exploration and exploitation is a primary factor in system survival and prosperity. (March, 1991, p. 71)

Another type of learning, evolutionary learning or better evolutionary algorithm, supports the creation of new generations of rules with help of cross-over or mutation. Cross-over "causes the characteristics of the parents to appear in new combinations in the offspring. It is the recombination of *sets* of alleles[9] that is most interesting from the point of view of rule discovery..." (Holland, 1995, p. 66) and mutation is "a process whereby individual alleles are randomly modified, yielding a different allele for the gene" (ibid., p. 70). With help of cross-over or mutation, it is possible to explore and discover new rules. These new rules will compete with the parent or previous rules in a selection process that determines if new rules are better adapted to the current situation. Such a selection process is often defined as a utility-based mechanism that tries to determine the best fit with the current environment.

Utility functions or utility-based mechanisms are often used as a performance measure—relative to a goal—for learning and can be distinguished into two types: (1) the agent has a model of the environment, which is used as an estimator for expected utility, e.g. cellular automata, and (2) the utility function is independent of the environment and choices are based on the utility of actions alone, e.g. Q-learning (Watkins, 1989).

Learning defines mechanisms that support the agent in the decision about what is wrong or what is right to do. However, the criteria for making the right decision are delivered by separate normative processes. These internal or external normative processes, i.e. internal drives or norms of a society, determine what goals the agent has to achieve and what behaviour is desired.

2.2.4.3 Goal-directed behaviour, meta-cognition and emotion

Goal-directed behaviour in combination with a performance system[10] allows the agent to learn and use the desired goal as a comparison to the current state of the agent. In SOAR, for example, every task of attaining a goal is formulated as finding a desired state in a *problem space*—a space with a set of operators that apply to a current state to yield a new state (Laird, Newell, & Rosenbloom, 1987). With the help of a goal, the agent can start to divide and conquer the problem[11] by introducing sub-goals—called universal subgoaling in SOAR—that are easier to fulfil than the main goal.

[9] Allele: one of two or more alternative forms of a gene that arise by mutation and are found at the same place on a chromosome (Pearsall, 2002).

[10] A performances system is: ... a system capable of certain performances [which is transformed by learning] into a system capable of additional [performances] (usually better ones; usually accomplished without losing much of the preexisting performance capability; and usually integrated with existing capability so they can be evoked on appropriate occasions). (Newell & Simon, 1972, p. 7)

[11] Among some of the methods used for solving problems are hill climbing, means-ends analysis and alpha-beta search.

The solving of problems requires goal generators[12] and goal comparators that are necessary to resolve conflicts (Sloman, 1987). These goal generators and comparators are driven by motives and constraints. Motives and constraints are often designed in a subsystem, e.g. a motivational, meta-cognitive, or emotional subsystem.

The motivational subsystem monitors drives (such as primitive drives, hunger, need for social interaction, etc.) and interactions that have impact on the selection of the goal and thereby the actions that satisfy the selected goal (Breazeal & Brooks, 2004; Sun, 2003). The meta-cognitive subsystem monitors and tries to improve the cognitive performance (Sun, Zhang, & Mathews, 2006).

Compared to the motivational system, meta-cognition is self-reflective; it can alter learning mechanisms or even switch to other tactics to improve performance.

The emotional subsystem perceives internal and external affordances. It regulates other cognitive systems, promotes appropriate decision-making and expresses the internal states to the outside world, e.g. emotional facial expressions (Breazeal & Brooks, 2004).

This outside world of an agent determines what is necessary to function in such a world. For instance, when the environment is populated with other agents, there can be a requirement to communicate in a language that allows agents to behave socially.

2.2.4.4 Environment, communication, language and social ability

The previous components were clearly fixed components of an agent. The elements mentioned here (environment, communication, language and social ability) are not explicit components of the agent, but rather influence the design of the components of the agent. For instance, agents that are not situated and not dependent on other agents and can solve their problems like chess computers, do not need language or social ability; they can solve problem spaces for a known pre-programmed accessible environment without 'really' perceiving the outside world. However, agents that live in an in-accessible environment surrounded with social (unpredictable) agents that require language to communicate with each other, depend on more than only a set of rules, i.e. they continuously have to adapt to new situations in a changing environment.

The definition of the complete agent with all its necessary components is always under debate. Many components will be discovered, removed or replaced in designing the agent, i.e. it evolves under the hands of an agent model designer. A nice bottom-up example of how an architecture can be constructed, see figure 2.3, is given in Autonomous Agents as Embodied AI (Franklin, 1997).

In figure 2.3, we can clearly see the complexity that occurs when we build a model out of just a couple of components. Therefore, researchers often put their efforts in certain parts of the architecture, e.g. perception has grown towards a specialised topic in cognitive science. Also there have been many discussions about whether a complicated agent, such as a cognitive architecture,

[12] Without goals the system is heading to an unknown direction and is not pro-active.

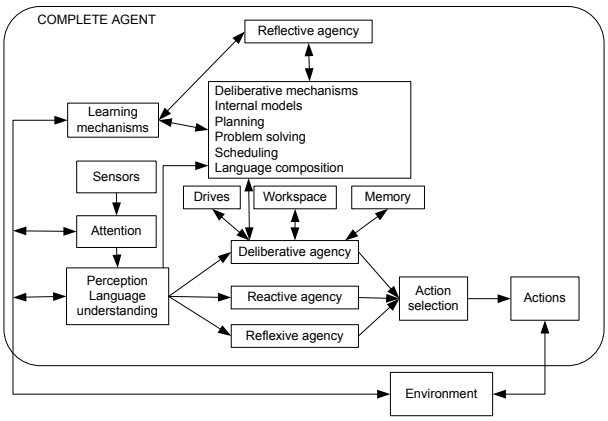


Figure 2.3: Agent with components (adapted from Franklin, 1997).

is really necessary, and feasible to implement in practical situations. These varying points of view have delivered many agent types that can be put together/constructed from a collection of components; even the way they are implemented varies. Among those variations we are interested in two types of agents: the *cognitive agent*, focusing on the internal components of the agent, and the *social agent* that focuses on the (social) interaction—cooperation and coordination—with other agents.

2.3 The cognitive and the social agent

With help of the presence (or not) of certain components, agents can be classified in different types. Many scientists give examples of classifications of agents (Genesereth & Nilsson, 1987; Jorna, 2002; Russell & Norvig, 2003; Wooldridge, 2002; Franklin & Graesser, 1996). The agents are often constructed of functional components, with each component fulfilling a specific function, e.g. vision, memory, motor control etc. In this section, we apply a classification commonly used in cognitive science and AI that builds up an agent from simple mechanisms towards complex physical and social situated mechanisms. Davis (2001) and Sloman (1993, 2001) have created such a classification of agent types: the reflexive/reactive, the deliberative and the reflective agent, comparable to the layers of agency shown in figure 2.3. This classification forms a basis under which all other cognitive approaches can easily find a home.

Two types of agent classification / agents are discussed in this section. The first is the cognitive approach (section 2.3.1 until 2.3.3) that varies from reflexive towards reflective. The second approach (section 2.3.4) will be the social agent. The cognitive agent is mainly concerned with the internal mechanisms of the agent, while the social agent is concerned with the influence of the environment affecting its behaviour and discusses aspects like autonomy, interaction with other agents, and normative and social behaviour.

2.3.1 The reflexive and reactive agent

Reflexive agents respond to the environment by an immediate action, whose functionality is often hardwired in the brain. Deliberation processes do not take place and the design of the model is often pure behaviour based, i.e. only a restrict set of known stimulus-response behaviours is implemented. The reactive agent is slightly more complicated. It gives structure to these behaviours and provides a bit more flexibility and control with help of mechanisms in order to adjust itself towards more different types of tasks. These agents do not have mechanisms that involve explicit deliberation or making inferences (a symbol system). They lack the ability to represent, evaluate and compare possible actions, or possible future consequences of actions (Sloman, 2001). For the reflexive agent, if necessary, a (static) goal can be set that states that the goal/behaviour, e.g. 'searching for food', becomes active as soon as the amount of energy in the body is at a too low level.

In case of the reactive agent: the subsumption architecture[13] of Brooks (1986) is an example of a reactive agent. The agent is physically situated and responds only to the current situation it gets involved in.

2.3.2 The reasoning agent

The reasoning or deliberative agent adds deliberative processing to the mechanisms used by the reflexive and reactive agent. The deliberative agent has many components in common with classical cognitive architectures (GOFAI[14]) and includes a representation of the environment, a memory, a workspace, a planning unit, management of goals and many other components that make process deliberation possible. According to Sloman (2001), a deliberative agent has a set of context-dependent and flexible processes, e.g. plan and goal generation, comparison, evaluation and execution providing the basis for a number of components, that cannot be handled by a purely reactive architecture. The interaction and the feedback that can be compared with the past representations stored in the memory give the deliberative agent opportunities to build up expectations of what effect certain actions will have on its environment and its own well-being. The ability to build up representations about its environment gives the agent the possibility to adapt and survive based on an experienced repertoire of the past. We can distinguish three types of agents that are commonly associated with the reasoning agent: the deductive reasoning agent, the practical reasoning agent (Wooldridge, 2002) and the cognitive plausible agent.

The deductive and practical reasoning agents often have a system that maintains a symbolic representation (logics) of its desired behaviour and a system that can manipulate this representation. Whereas the deductive reasoning agent works with deduction and theorem-proving, the practical reasoning agent is specialised in reasoning that is directed towards the future in which selection between conflicting considerations are provided by the agent's desires/values/cares and what the agent believes (Bratman, 1990). A well known

[13] We will give an explanation of the subsumption architecture in section 4.1.2

[14] Good Old Fashioned AI

implementation of this agent is the beliefs/desires/intentions (BDI) architecture or Procedural Reasoning System (PRS) (Wooldridge, 2000). The weakness of these agents is that although they form representations, such as beliefs, desires and intentions, they only operate at the intentional level (Dennett, 1987), and therefore developers of these agents are not concerned with how these agents could be symbolically grounded. Or as Wooldridge states: "... I will not be concerned with *how* beliefs and the like are represented... the assumption that beliefs, desires, and intentions are symbolically represented is by no means necessary for [the modelling of BDI agents]" (Wooldridge, 2000, p. 69).

In contrast to the deductive and the practical reasoning agents, the cognitive plausible agent operates at the intentional *and* functional level, and is based on more than ascribing mental attitudes to agents alone. The agent is said to be cognitive plausible when it has a cognitive architecture not only based on a physical symbol system (Newell, 1980), cognitive mechanisms, goal-directed behaviour and learning capabilities, but is empirically tested as well (cf. Newell, 1990; Anderson, 1990; Van den Broek, 2001), i.e. the definition of a cognitive plausible agent is:

> [A *cognitive plausible agent*] is a goal-directed decision-maker who perceives, learns, communicates, and takes action in pursuit of its goals, based upon [a physical symbol system] that implements theories of cognition [and is supported by empirical evidence]. (Van den Broek, 2001, p. 88)

The introduction of production systems (Newell, 1973) as systems that can implement theories of cognition are adopted by many researchers in cognitive science. A production system is a physical symbol system that exists out of a set of productions (condition-action patterns) and a set of data-structures. A set of goals in combination with means-end reasoning allows the agent to explore problem spaces and exhibit intelligent action (Newell & Simon, 1972). SOAR (Newell, 1990) and ACT-R (Anderson & Lebiere, 1998) are well known examples of agents with a cognitive architecture. These systems can store and manipulate representations and contain subsystems and mechanisms that enable the actor to adapt (e.g. sub-symbolic learning in ACT-R) and learn (e.g. chunking in SOAR). For more details, we refer to chapter 4 that discusses cognitive theories and gives an elaborate description of a cognitive plausible agent.

2.3.3 The reflective agent

The reflective agent builds on top of the previous agent. It is equipped with a meta-management module that observes and monitors its own cognitive processes in order to reach better performance that serves the goal or objective the agent has set its mind to. For instance, the reflective agent can for instance use the following operations for monitoring its own cognitive processes (Sloman, 2001): (1) the ability to think about and answer questions about one's own thoughts and experiences, (2) the ability to notice and report circularity in one's thinking, and (3) the ability to notice opportunities for changing one's thinking.

Sun et al. (2006) term influencing the cognitive processes as meta-cognition and implement this in the cognitive architecture CLARION (Sun, 2003, 2002). They give examples of mechanisms that aim for—to name a few— (1) behaviour monitoring, e.g. setting of goals and reinforcement functions, and (2) information filtering, acquisition and utilisation of different learning methods and reasoning methods. Research at this layer has just started and the addition of reflection creates more complexity, especially when an agent is equipped with many components that can be reflected upon.

Another example of reflection is the introduction of emotional aspects that can drive the agent in changing its behaviour radically. Such changes can be caused by properties that say something about the current status of the system, e.g. the agent feels itself bored, angry or afraid.

The focus of this dissertation is not to include the reflective mechanisms of the individual, but is more directed at reflection by interaction with the outside world and other actors. In the next chapter we will introduce the social construct that operates at the normative level of an agent and can influence the production system in its (performance) outcomes (e.g. allowing or prohibiting certain outcomes).

2.3.4 The social agent

Another approach of looking at agent typology is to incorporate the social and organisational aspects of the agents. For instance, the social and cognitive agent of Conte and Castelfranchi (1995b) and the Computational & Mathematical Organization Theory (cf. Carley & Gasser, 1999) highlight the importance of agents that create social goals and are physically, socially and culturally situated as well. Verhagen (2000) defines a typology (cf. Conte & Castelfranchi, 1995b) of autonomous agents that have their own goals, in the pursuit of which they might refuse to do what they are asked to do.

1. Reactive agents
   Comparable with the cognitive typology: a reactive agent has no means of influencing the environment in a preconceived way. The autonomy resides completely in the combination of environmental cues and properties of the system.

2. Plan autonomous agents
   Plan autonomous agents have more autonomy than reactive agents in the sense that they may choose how to achieve a certain state of the world. They still do not have the notion of goal, i.e. their goals are determined by a control system from the outside. A given goal can be connected to a repertoire of actions that they can aggregate into a sequence resulting in a plan that will achieve the objective, i.e. the agent has a plan library that is matched with a given goal.

3. Goal autonomous agents
   Goals are not just created by requests but must also be linked to goals the

agent already has. A goal autonomous agent has the autonomy to determine what its "prevailing interest" is, considering its goals. It will judge which states of the world are its interests by evaluating what it provides in terms of degree of goal satisfaction and goal priority.

4. Norm/social autonomous agents
These agents choose which goals are legitimate to pursue, based on a given system of norms. The agent has the autonomy of generating its own goals and is able to choose which goal it is going to pursue. Besides that, the agent is equipped with the capability to judge the legitimacy of its own and other agents' goals. When a goal conflict arises, the agent may change its norm system thereby changing priorities of goals, abandoning a goal, generating a new goal, etc. Norm autonomous agents generate norms they can use to evaluate states of the world in terms of whether or not they could be legitimate interests. Legitimacy is a social notion and is in the end determined by the norms of the agent with respect to the agent society it is part of.

Right now, we do not want to explain exactly what is necessary for constructing a social agent, but only show a typology of agents. However, we will return to social capabilities in section 2.5 that is concerned with coordination and in chapter 3 that elaborates about the characteristics of the social actor.

2.3.5 The complete cognitive and social agent

In this section we want to reflect on the previous typologies of the cognitive and social agent and try to map their characteristics or capabilities next to each other and thereby see what differences there are.

Carley and Newell (1994) made an attempt to determine the characteristics of a cognitive and social agent. They have created the so-called Model Social Agent (MSA) which is in our opinion a complete social and cognitive agent. Their description of the Model Social Agent targets quite well the type of agent we are looking for in our research.

> The Model Social Agent has information-processing capabilities and knowledge. Agents' information-processing capabilities are goal oriented. They control the agent's ability to handle information. Agents exist within an environment which is external to the agent's processing capabilities. The agent's knowledge is to an extent dictated by the external environment in which it is situated. The Model Social Agent exists in a particular situation (both physical and social). This situation is the environment perceived by the agent, but how the agent encodes it, and how much of the environment is encoded by the agent is an open issue[15]. The agent has a goal. The agent enters a situation with prior knowledge. The agent may have

[15] In this dissertation we will try to fill in this open issue with the introduction of the social construct in the next chapter.

> an internal mental model of the situation that differs from that held by other agents in the same situation. Throughout, we take the agent as having the typical human sensory and motor devices to sense the environment and affect the situation. We are concerned with the nature of the inner system that makes interaction social... (Carley & Newell, 1994, p. 223)

Carley and Newell describe the agent along two dimensions: increasing limited *processing capabilities*—limitations that enable it to be social, and increasing rich situations—*richness* of the agent's *perceived environment* that evokes and supports social behaviour, i.e. how the agent is situated in the environment. The dimensions give the possibility to create a 1x1 classification matrix (Carley & Newell, 1994, p. 243) along the lines of limited processing capabilities and the limitation in richness of the perceived environment.

Along the line of limited processing capabilities, five types of agents can be discerned:

1. The Omnipotent Agent (OA): the agent "knows all there is to know about the task environment in which it is embedded" (ibid., p. 227).

2. The Rational Agent (RA): "An agent has its own body of knowledge and it behaves rationally with respect to that knowledge by taking those actions that its knowledge indicates will lead to attaining its goals" (ibid., pp. 227–228).

3. The Boundedly Rational Agent (BRA): The boundedly rational agent has ecological and cognitive limitations of both (environmental) knowledge and processing capability (Simon, 1945), i.e. the agent "has limited attention and therefore cannot process all the knowledge available in its task environment" (ibid., p. 228).

4. The Cognitive Agent (CA): "The cognitive agent is the boundedly rational agent with a fully specified architecture... Where the boundedly rational agent is a set of general claims, the cognitive agent is a set of specific operational details. Moving from general principles to specific architecture further limits the agent. Where the boundedly rational agent may have some type of knowledge structure called a chunk, in the cognitive agent we know the exact form of these chunks and the procedure for creating them" (ibid., p. 230).

5. The Emotional Agent (EA): "The emotional agent, while more human than the cognitive agent, also is more constrained in its ability to attain its goals. We could add emotions to each of the other limited agents, the rational, boundedly rational and cognitive agents. These all would reflect the same notions of how emotions limit goal attainment" (ibid., p. 235).

Carley and Newell also made a distinction along the other dimension that varies in the richness of the situation the agent perceives, i.e. the types of knowledge available to the agent determines the types of behaviours the agent can

engage. They have distinguished five[16] types of situations the agent can be situated in.

1. Non-social Task Situation (NTS): The non-social task situation is a situation in which an agent only deals with a task environment that is not affected by social factors from other agents, i.e. in the extreme case, we can take the chess computer that only deals with the outcome of the chess-game. The chess agent deals with a fixed physical environment and deals with the moves of a player, but does not take into account any social factors of the player that plays against him.

2. Multi Agent Situation (MAS): In this situation, the agent deals with other agents and treats them as having their own goals, but the goals are not social goals that maintain social relations.

3. Real-time Interactive Situations (RIS): The situation is similar to the previous situation, but the agent has to balance the processing of interpreting information versus acting with its environment as well as responding to its environment in a timely fashion.

4. Social Goal Structure Situation (SGSS): "This situation is characterized by the presence of groups and group related knowledge such as rules for membership, maintenance, and dissolution... The existence of social structure permits the separation of task and self; i.e. the agent may now be faced with goals that are demanded of it by an external entity, such as a group or organization. The agent, situated in a social structure, may now be faced with multiple goals[:]... (1) task-related goals, (2) self-maintenance and enhancement goals, and (3) social-maintenance goals... that may compete and conflict with each other, as well as multiple goals at the same level" (ibid., pp. 239–240).

5. Cultural-Historical Situations (CHS):

   > The agent exists at a particular point in time and history, in a particular society, at a particular place, with particular institutions, and so on. The situation is affected by, and has resulted from, a historical process that led to the current state and moment. The agent has available a body of social practice that has been shaped by a specific culture in the ways that may be idiosyncratic to that culture. The agent has "developed" within this situation and so is socialized to a particular way of doing things, to specific beliefs, norms, and values. (ibid., p. 239)

Carley and Newell created the "Model Social Agent" as a combination of the Cognitive Agent (CA) and the characteristics of the emotional agent (EA)

[16] Instead of the six types that Carley and Newell distinguish, we have merged two types or situations into one, i.e. we have merged Social Goals Situation (SGS) and Social Structure Situation (SSS) into Social Goals Structure Situation (SGSS), because they are, according to their description, almost similar.

that deals with other agents in a cultural-historical situation (CHS). With help of the dimensions, they have tried to create a better understanding of the social nature of human beings and applied it as a framework in order to determine how far current research is and what the aim of that research should be to reach the "Model Social Agent".

We adopt both dimensions to determine what type of agent and environment, as suggested by Carley and Newell, fits the characteristics of the cognitive and social agent in the previous sections. Table 2.1 gives an overview of the characteristics of the cognitive and social agent and is an estimation of the capabilities of the agents[17].

Referring back to research question 1 (particular questions 1.1. and 1.2) and after examining table 2.1, we can state that in order to create an agent that is both cognitive and social, it requires a combination of the characteristics of the cognitive plausible agent and the norm/social autonomous agent. Therefore, we will address the social agent in chapter 3, and the cognitive agent in chapter 4 in order to create a combined—social and cognitive—agent that conforms as much as possible to the Model Social Agent.

The discussion in this section has elaborated that the notion of a social agent can only exist in an environment with other agents and moreover that bounded rationality creates at the same time the need for that interaction. This need can *only* be fulfilled if the environment is possible to deliver such interactions. Therefore, a well designed model of an agent should take into account the properties of the environment in order to make sure that coherence between environment and agent is optimal.

[17]The table is an estimation, in the sense that some agents can have more or less characteristics than shown in the table, i.e. the classification of the agents is not as strict as shown in this table, but is more a gliding scale or approximation.

	Level of description	*Production or plan oriented*	*Goal / behaviour oriented*	*Neural plausible*	*Meta-cognition / motivation*	*Mutual Modelling*[a]	*Physically situated*	*Communication (language)*	*Socially situated*	*Processing capability*	*Richness environment*
Cognitive agent											
Reflexive / reactive	B			-	-	-	+	-	-	OA	NTS
Deductive Reasoning	R	Pl		-	-	-	-	-	-	OA	NTS
Practical Reasoning	R	Pl	G	-	-	+/-	+/-	+/-	-	RA	MAS, RIS
Cognitive plausible	R,F	Pr	G	+	+/-	+/-	+/-	+/-	-	BRA, CA, EA	MAS, RIS
Social agent											
Reactive	B			-	-	-	+	-	-	OA	NTS
Plan autonomous	R	Pl		-	-	-	-	-	-	OA	NTS
Goal autonomous	R	Pl	G	-	-	+	+/-	+/-	-	RA	MAS, RIS
Norm / social autonomous	R,S	Pl	(S)G	-	-	+	+/-	+	+	RA	MAS, RIS, SGSS
B = Behaviour R = Rational F = Functional S = Social	+ = yes +/- = possible - = no	Pl = Plan Pr = Production G = Goal								OA = Omnipotent Agent RA = Rational Agent BRA = Boundedly Rational Agent CA = Cognitive Agent EA = Emotional Agent	NTS = Nonsocial Task Situation MAS = Multi Agent Situation RIS = Realtime Interactive Situation SGSS = Social Goal Structure Situation	

Table 2.1: Characteristics of the social and cognitive agent.

[a]Mutual modelling is the ability of an agent to build up a representation of the representations of other actors, e.g. I believe that he believes. We refer to section 2.5 for more details about mutual modelling.

2.4 The environment of the agent

The properties of the environment determine partly the constraints and the properties of the model of the agent. Russell and Norvig (2003) define a list of dimensions along which environments can be categorised.

- Fully observable vs. partially observable
  A task environment is fully observable if the sensors of the (omnipotent) agent detect all aspects that are relevant to the choice of action. On the other hand, in a partially observable environment, the agent has no access to all information that is hidden in the environment or other actors.
- Deterministic vs. stochastic
  When the agent can influence and create a next state in the way it wants, the agent is dealing with a deterministic environment, i.e. during the time the agent is deliberating, the environment does not change. An environment that consists of more agents and is partially observable is defined as being stochastic. The agent lives in an environment that continuously changes; behaviours of other agents become more or less unpredictable.
- Episodic vs. sequential
  The time line in an episodic environment is divided into different episodes that do not influence each other. An example of an agent that acts in an episodic environment is an agent at an assembly line in a factory that repeatedly mounts a part on a car in which the current decision does not affect future decisions, i.e. in the case of the assembly line, the environment returns to its begin state. Therefore, the learning capabilities in these types of environments are small. On the other hand, in sequential environments, it is possible for the agent to learn, explore and reason about future states.
- Static vs. dynamic
  The environment is dynamic when it develops while the agent is deliberating about the environment. A static environment, such as a chess game, does not change during the time the agent is calculating the next move.
- Discrete vs. continuous
  The discrete-event simulation and finite state machines are examples of discrete environments and the time in these environments evolves step by step. Whereas in a continuous environment the agent has a continuous-state and continuous-time problem, e.g. a real-time system[18] in an aircraft that controls the plane.
- Single agent vs. multi-agent
  A Multi-Agent System compared to a single agent is complex, caused by interaction that occurs between agents. Interaction in the form of communication can lead to organisational and social behaviour, conflicts or cooperation resulting in complex behaviour.

[18] Neglecting the fact that the processors of most machines are time-slicing and do divide any simulation in time steps

The environment has impact on the autonomy of an agent. Odell, Van Dyke, Fleischer, and Breuckner (2002) define an environment[19] as follows: "An environment provides the conditions under which an entity (agent or object) exists" (p. 2). They define three types of environment:

The physical environment: provides the principles and processes that govern and support a population of entities.

The communication environment: provides those principles, processes, and structures that enable an infrastructure for agents to convey information.

The social environment: a communication environment in which agents interact in a coordinated manner.

The communication environment cannot exist without a (virtual) physical environment and the social environment cannot exist without the communication environment, see figure 2.4.

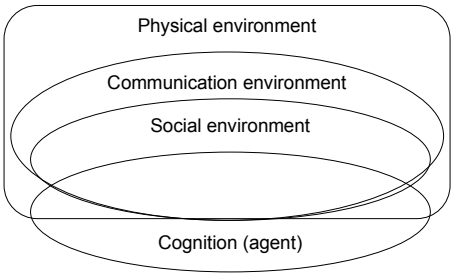


Figure 2.4: Nesting of environments (adapted from Odell et al., 2002).

The physical environment is often *simulated*, because simulated environments give the ability to easily create, change or test many different situations in which the agent may be involved. On the other hand, robotics is a field that tests agents in a real physical world, e.g. the laboratory in which the circumstances are still under a certain control. The simulated physical environment has to do the work that the real physical world does in the case of real-time systems: enforce the consistency in time, in space, and with respect to physical laws, i.e. it has to maintain synchronisation between agents and environment. In this respect, a simulation has to do more work (not less) than a system functioning in the real world. The same can apply for the communication and social environment. Communication and interaction in a MAS simulation are still under control of the experimenter, while HCI (Human-Computer-Interaction) tries to connect to the real interactive and social world by creating a mixed environment of humans and computers. Another important environment is the task environment. The task environment can be seen as an environment that represents tasks that are possible for an agent under restriction of the physical, communication and social environment.

[19] In the next chapter, we explain the 'semiotic Umwelt'; an environment that consists of signs and enables the agent to interact, i.e. the agent perceives, interprets and acts upon those signs.

Task Environment

The way tasks can be performed and what kind of tasks can be performed is constrained by the physical, communication and social environment. These environments construct a set of possibilities for the agent to respond to, i.e. the outer task environment. However, the agent has attention for only a subset of tasks in the outer environment[20] and this specific set is an unique task environment to the agent himself, i.e. it is the combination of the inner and outer environment that is relevant for an agent to do a task. Simon also addresses the distinction between an inner and an outer environment:

> The advantage of dividing outer from inner environment in studying an adaptive or artificial system is that we can often predict behavior from knowledge of the system's goals and its outer environment, with only minimal assumptions about the inner environment.... In one way or another the designer insulates the inner system from the environment, so that an invariant relation is maintained between inner system and goal, independent of variations over a wide range in most parameters that characterize the outer environment. (Simon, 1996, p. 8)

The difference between inner and outer environment is evident when we compare the reactive agent (embodied cognition) with the cognitive agent (classical approach). The reactive agent its task environment is mainly situated in the outer environment and is (re)constructed the moment the environment changes. On the other hand, the classical approach represents the majority of tasks as problem spaces in the mind of the agent self. Embodied cognition or situated cognition have stressed that there is clearly an interdependent relationship between the internal represented task environment and the physical, communication and social environment, i.e. the model of the agent can constrain for instance actions of other agents (the social environment), and the physical environment can constrain the options (tasks) of the agent.

The tasks that an agent has to accomplish in its environment create (partly) the need for a situated model of the agent that has to fulfil the requirements of the environment. For instance, the environment of a chess computer can be the chess computer itself, getting input from the position of the pieces on the board and its internal rule-base. Whereas an agent that guides a plane to land is more situated in the environment and has to respond quickly to changes in the environment, e.g. side-winds etc. The same aircraft agent does not only have to be communicative, but social (and economic) as well when it has to interact with other planes to negotiate for a time-slot to land the plane.

In our research, we are interested in the aspects of actors that explain social behaviour. Therefore we need a complex environment (see chapter 5), i.e. a physical, communication and social environment. This environment is a specific environment suited for the traffic experiments conducted in chapter 6. The environment is a relatively rich environment, comparable to the SGSS (Social Goal Structure Situation) that consists of other agents, a communication environment

[20] The bounded rationality argument (Simon, 1945).

and actors that have the capability to create a social environment. The purpose of the environment is to create the possibility for simulating (social) interaction and coordination (mechanisms) between actors that have to solve (shared) problems. The next section will discuss conflicts, shared problem solving between interacting agents and coordination mechanisms that try to reduce the conflicts between agents.

2.5 Agent and Coordination

Partly due to the inaccessibility of the environment, the life of a human being and its (knowledge) resources are scarce, therefore agents are boundedly rational (Simon, 1945). Hence, the need for cooperation and conflict in order to survive is inevitable. In the first part, we will elaborate when coordination occurs and what the motives[21] are to cooperate and solve conflicts. In the second part of this section, we will discuss several coordination mechanisms that try to avoid conflicts and coordinate problem solving.

2.5.1 Cooperation and Conflict

Conflict or cooperation suggests that there are relationships between activities that, according to the coordination typology (see figure 2.5 of Von Martial (1990), either are negative relationships (conflict) or positive relationships (cooperation).

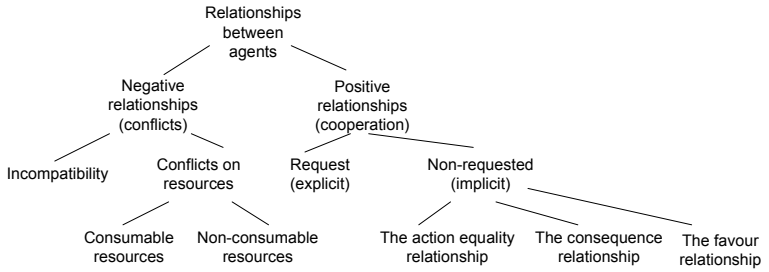


Figure 2.5: Typology of coordination relationships (Von Martial, 1990).

The negative relationship occurs when agents have an incompatibility between goals. For example when two agents execute actions that lead to a goal, but the outcome of the success (the goal) conflicts with the other, e.g. in a car race, only one car can win. The same can happen with resources that are shared

[21] Economic motives are important for estimating if cooperation is fruitful. Coordination costs are not explicitly discussed in this section, but for example, they can exist of "(Jorna, Gazendam, Heesen, & Van Wezel, 1996, p. 29): (a) organization, including establishing contracts with organizations that perform work that is contracted out, (b) improvisation, (c) slack capacity of actors, means of production and materials in stock, (d) making plans, (e) replanning and adjusting plans, (f) storing and communication of plans, including the costs of maintaining an information and communication system, and (g) accepting plans by the actors that have to perform the plans as well as by other stakeholders" (as cited by: Gazendam, 2006, pp. 167–168).

by agents, e.g. occupying a parking spot. The positive relationship is one in which there is either an explicit request (e.g. speech) for cooperation or there is a non-requested relationship that turns out to be favourably for one of the agents and is not harming the other.

Von Martial (1990, p. 112) distinguishes three forms of non-requested relationships:

1. The action equality relationship. We both plan to perform an identical action, and by recognising this, one of us can perform the action alone and so save the other effort.

2. The consequence relationship. The actions in my plan have the side-effect of achieving one of your goals, thus relieving you of the need to explicitly achieve it.

3. The favour relationship. Some part of my plan has the side effect of contributing to the achievement of one of your goals, perhaps by making it easier (e.g. by achieving a precondition of one of the actions in it).

In a similar vein, following Schmidt (1991), Gazendam and Homburg (1996) distinguish four forms of cooperation.

1. Augmentative cooperation. When single agents are limited by their physiological and mechanical capabilities, and physical or economic resources, cooperation allows agents to achieve a task which would otherwise be impossible.

2. Integrative cooperation. The task requires specialization from agents in order to achieve it, because specialization decreases throughput time and reduces a complex problem to smaller and simpler problems that can be solved by individual agents.

3. Debative cooperation. Agents bring in a variety of interests and values, and aim to come to a compromise in which every agent takes a position and is a process in which agents want to come to a mutual agreement.

4. Conflict handling. The handling of conflicts can be controlled by an authoritative organ or 'rules of the game' that prevent agents from ending up in a deadlock or destructive conflict.

The first two types of cooperation can clearly be nested under the positive relationships and fall in a similar category as the requested or non-requested relationships of Von Martial. The last two types are actually conflict and cooperation situations at the same time. The agents start of with conflicting interests, however debative cooperation transforms the initial conflict situation into a cooperation situation. The same applies for conflict handling; the conflict situation is oppressed by authority, norms and rules causing a potential conflict situation transform into a—possible (in)formally forced—cooperation situation.

Cooperation and prevention of conflicts urges the agent to communicate with others, be it verbal or non-verbal. Communication that is expressed in an

(mutually) agreed language gives the possibility to express and transfer meaning of complicated beliefs, goals and so on, to other agents. The possibility of understanding each other's needs gives the drive to social behaviour and the possibility for coordination. In society, coordination is often linked with conflicts about scarcity, i.e. many situations can occur in which scarcity is a reason for conflict.

Ferber (1999) assigns three criteria that determine the type of conflict or cooperation situation that can take place. In the first place, when agents are in an environment, their goals can be compatible or incompatible, i.e. cooperation vs. antagonism. Secondly, goals[22] are often connected to resources: when two agents want access to the same resource, a conflict may arise and coordination may be necessary to solve the problem. Thirdly, to achieve goals, they need to be linked to procedures or skills. Some agents do not have the required skills and need skills of others to achieve their goal. Based on these three criteria, a typology of different situations can be made explicit, see table 2.2.

Goals	**Resources**	**Skills**	**Types of situation**	**Cooperation type**
Compatible	Sufficient	Sufficient	Independence	
Compatible	Sufficient	Insufficient	Simple collaboration	Integrative
Compatible	Insufficient	Sufficient	Obstruction	Augmentative
Compatible	Insufficient	Insufficient	Coordinated collaboration	Augmentative Integrative
Incompatible	Sufficient	Sufficient	Pure individual competition	Debative Conflict handling
Incompatible	Sufficient	Insufficient	Pure collective competition	Integrative Debative Conflict handling
Incompatible	Insufficient	Sufficient	Individual conflicts over resources	Augmentative Debative Conflict handling
Incompatible	Insufficient	Insufficient	Collective conflicts over resources	Augmentative Integrative Debative Conflict handling

Table 2.2: Classification of interaction scenarios (adapted: Ferber, 1999)[23].

[22] Goals sometimes are seen as resources, (e.g. 'That is my goal!') however there are many routes that end up in Rome, in which resources (roads, cars) are connected to the goal. Skills are also often seen as a resource (e.g. an agent is a resource), however here we distinguish it as a mean-to-an-end.

From the typology, it becomes clear that interaction situations become more complicated when goals start to conflict, resources become limited and skills are not adequate. In the optimal situation (Independence) there is indifference about what any agent does, i.e. there is no scarcity. Situations, for which the agents have shortage of skills, compensation in the form of Integrative Cooperation is necessary. In the case of insufficient resources, agents are dependent on each other. Augmentative Cooperation can reduce the lack of physiological, mechanical, physical or economic resources.

Although in situations where goals are compatible *and* there are either insufficient resources or skills in which case augmentative or integrative cooperation can solve coordination problems, negotiations will always remain necessary to make sure that there is (mutual) agreement between agents.

In situations where goals are incompatible, a deliberative form of cooperation like debative cooperation in combination with conflict handling, is necessary. Incompatibility can occur when there are different interests in an organisation. For instance, the tension between the marketing department and the production floor. The marketing department tries to sell as much as possible, while the production floor tries to make sure that the marketing department does not create goals that are unable to fulfil.

Thus, solving conflicts and cooperation requires coordination between agents in order to achieve goals by negotiation, allocating resources and skills. Coordination mechanisms facilitate coordination and are discussed in the next section.

2.5.2 Coordination mechanisms

Coordination mechanisms are mechanisms that stabilise and better control issues concerning scarcity and aim at achieving coordinated action. Economic science (the economic perspective, see next section) has grown out to be an important science in studying scarcity issues. They attempt to regulate our economy by the value of capital and market mechanisms. These mechanisms are expected to control the way individuals should behave, thereby coordinating and regulating the actions of individuals in a structured way. However, economists study the population at a relative high level of abstraction and when we look at the economic situation today, market mechanisms as coordination mechanisms do not achieve what they are expected to do. An alternative for understanding coordination is given by Distributed Artificial Intelligence (DAI) (section 2.5.2.2).

[23] **Incompatibility of goals:** In the first place, incompatibility of goals can occur. It creates a situation in which agents conflict with each other in order to achieve a successful outcome. As mentioned before in a car race, only one can win.
Insufficient resources: The agents are not directly dependent on each other, i.e. the actions and goals are indirectly connected by the resources each agent wants to use. Two types of resources can be distinguished, resources that are used for temporary use and resources that are consumed and change ownership. In the first case, resources can be allocated with help of planning and division of tasks, i.e. first one agent uses the resource and then the other. In the second case, the resource is consumed and can only be allocated to one agent.
Insufficient skills: When an agent has not the capability of achieving a goal it can solve this shortage by outsourcing a task to another agent—when available—or receive training from other agents and thereby has built up experience when the problem reoccurs.

DAI, in cooperation with for instance economics, psychology and sociology, applies sophisticated methods and models to understand how individuals behave.

2.5.2.1 The economic perspective

Economists often take the market or bidding systems as coordination mechanisms that economise and disclose information from bidders and sellers. The actors in this approach are often companies and the market functions as a price system and prices are a way to inform the 'rational' actor about the equilibrium in the market. Prices or utility mechanisms give the ability to 'play games' and try to find an optimal (minimax) strategy to maximize a safe outcome. To describe 'gameplay', game theory is widely adopted in economics and also has found its way into MAS. Game theory (Axelrod, 1984; Morgenstern & Von Neumann, 1947) is applied in studying interaction—conflict and cooperation—and enables the researcher to analyse and understand strategic games or scenarios. Game theory applies a payoff matrix in which the agent has a self-interest that can be expressed with the help of utilities. Table 2.3 displays such a matrix: the numbers reflect the utility both players assign to the outcome if they choose for a certain strategy.

Player I / ***Player II***	Cooperates	Defects
Cooperates	3 3	5 0
Defects	0 5	2 2

Table 2.3: Strategic form of game theory; the prisoner's dilemma.

The matrix presents a popular example of game theory, i.e. the prisoner's dilemma (Axelrod, 1984). The scenario can be described as follows. If one of the prisoners confesses (defects) and the other not (cooperates), then the confessor will receive a reward by giving him less punishment and giving the other additional punishment compared to the standard. However, if both confess (defect), then they get both the standard punishment or if both don't confess (cooperate) they don't get any punishment at all.

Game theory assumes a varying surplus of cooperation and scenarios are often a dichotomy of actions such as in favour of, or against. It is very useful and shows the possibilities of agents in an easy to understand payoff matrix and is applicable in MAS as a *descriptive* theory, i.e. it is a high-level theory (rational level) that neglects underlying mechanisms.

There are however some counter arguments for the game theory (Wooldridge, 2002). First of all, people are not always self-interested, and also show

forms of altruism and spontaneity. Secondly, human-beings are not supposed to be totally rational and, hence, have not perfect information of the environment they live in. The third problem is that the game theoretical approach is often played with two players and does not take into account past experiences, i.e. the so called one-shot games. The Iterated Prisoner's Dilemma (IPD) of Axelrod (1984) introduced the concept of a Multi-Agent System in which every agent has prior knowledge about the step its opponent did before and selected 'cooperate' or 'defect' based on this step. The introduction of iteration and many other techniques makes game theory worthwhile to apply in Multi-Agent Systems for estimating and describing how a system might stabilise at its equilibrium points (cf. Nash, 1950).

Still, in game theory, the agent does not account for social action aimed at modifying the behaviour and mental states of the other agents. Game theory gives no explanation for why agents engage in social action and what internal structures make social interaction possible. Also no explanation is given of how the behaviour at the macro-social level influences the internal mechanisms of the agents (Conte & Castelfranchi, 1995b). Compared to the economic perspective, the DAI perspective tries to incorporate the needs of the individual and the society as well.

2.5.2.2 The DAI perspective

The DAI perspective[24] is a more detailed approach compared to the economic perspective, i.e. coordination is achieved by decomposing the organisational entity into manageable units, such as tasks, goals and actions. In order to control coordination, the process of managing interdependent activities is crucial (Malone & Crowston, 1991). DAI is concerned with coordination behaviours or mechanisms that take care of managing conflicts and cooperation.

> Coordination behaviors [or mechanisms] can be divided roughly into specification behaviors (creating shared goals), planning behaviors (expressing potential sets of tasks or strategies to accomplish goals), and scheduling behaviors (assigning tasks to groups or individuals, creating shared plans and schedules, allocating resources, etc.). Coordination behaviors in turn rest on more basic agent behaviors such as following rules [norms and social laws], creating organizations, communicating information, and negotiation or other conflict resolution mechanisms. (Decker, 1995, p. 9)

In this section, we will shed light on coordination mechanisms that try to solve problems of cooperation and conflict.

[24] As mentioned before, Multi-Agent Systems and Distributed Problem Solving are part of DAI. There is considerable overlap between MAS and DPS, therefore in this section there is not made a clear distinction between them. However, sections 2.5.2.2.1 until 2.5.2.2.3 can be seen as falling in the category of DPS, while the remaining sections have more characteristics from MAS.

2.5.2.2.1 Cooperative Distributive Problem Solving

Cooperative Distributive Problem Solving (CDPS) tries to coordinate the relationships between agents in order to cooperate and solve problems. In CDPS, the following assumptions are made (Durfee, 1999). There is already a certain group coherence, i.e. agents have been designed as a group and agents need to know how to work together. Besides that, the agents are assumed to be benevolent. Based on these assumptions, a joint plan can be constructed in which task allocation can take place.

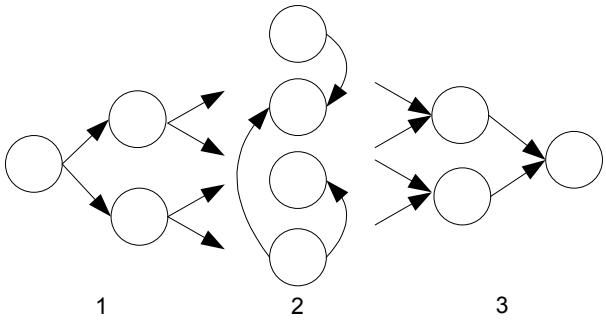


Figure 2.6: The three stages of CDPS (Smith & Davis, 1980; Wooldridge, 2002).

Task sharing in CDPS exists out of several stages (Durfee, 1999), see figure 2.6. A complex task can often be subdivided (1) into several tasks that (2) can be distributed among other agents. In the stage of solving the sub problems, agents solve their own problem and if necessary share knowledge with others to solve the problem. When the sub-problem is solved, the agent sends its solution to a manager agent (3) who oversees the whole problem and solves the main problem. Although simple problems can be solved efficiently and effectively in this way, in heterogeneous systems, where agents are trained as specialists, solving problems and scheduling jobs with agents becomes rather complex. A solution for this problem is a communication protocol, such as Contract-Net (Smith & Davis, 1980). With help of an Agent Communication Language it is possible with CNET to set up an (economic) market principle of bidding and selling in which managers try to find contractors (who can try to find sub-contractors). The process involves the announcement of tasks to others. The others can give a bid on the task and after some time contractor(s) are selected by the manager. When agents are confronted more than once with a problem to solve, then agents can try to improve their performance by sharing the results of sub-problems. Sharing provides for error checking, and increased confidence and precision, a better overview of the overall problem (completeness) and solving the overall problem faster (Durfee, 1999). However, coordination problems can be structured and solutions can be achieved faster and more properly with the help of planning and scheduling.

2.5.2.2.2 Planning

When in Multi-Agent Systems tasks of an agent depend on other agents' uncertainty increases and coordination becomes necessary. Planning and scheduling can release this tension and supports the increase of performance—efficiency and effectiveness—and possibly reduces uncertainty and gives more self-knowledge of the task-structure.

In a Multi-Agent System, planning is characterised by four aspects (Gazendam, 2006, p. 162):

1. Multi-agent plans are distributed, which means that they generally are represented at many places: in the minds of agents and in the documents that agents use.
2. The planning process aims at creating or maintaining coordinated behavior or social order by creating social constructs (see chapter 3) in the form of plans.
3. The planning process is characterized by problem solving intertwined with negotiation. In this process, social constructs (plan elements) emerge out of solving conflicts.
4. In the planning process, actors determine the acceptability of a proposed course of action by estimating their share in the costs and benefits of the resulting cooperative activity.

In DAI, a similar approach, i.e. Partial Global Planning (PGP) (Durfee & Lesser, 1991) is taken by addressing the resources (including goals and skills) as nodes.

> [PGP is] a flexible approach to coordination that does not assume any particular distribution of subproblems, expertise, or other resources, but instead lets nodes coordinate in response to the current situation. Each node can represent and reason about the actions and interactions of groups of nodes and how they affect local activities. These representations are called **partial global plans** (PGPs) because they specify how different parts of the network plan to achieve more global goals. Each node can maintain its own set of PGPs that it may use independently and asynchronously to coordinate its activities.
>
> - PGP avoids redundant work among nodes by interacting among the different local plans.
> - It schedules the generation of partial results so that they are transmitted to other nodes and assist them at the correct time.
> - It allocates excess tasks from overloaded nodes to idle nodes.
> - It assumes that a goal is more likely to be correct if it is compatible with goals at other nodes.
>
> (Decker, 1995, pp. 33–34)

Thus, the first step is that each agent—able to explicitly represent its goals and actions—constructs its own local plan. The agent has no knowledge of other agents' plans and first needs to create its own plan and uses scheduling techniques in order to clarify its actions—when, what, where, etc.—to others. The next step is to communicate and exchange information between agents, followed by a reordering of the local plan and storing of (intermediate) agreements or results in the partial global plan. This process can be iterated until the joint goal is solved or desired situation is reached. The explicit modelling of teamwork is another approach for creating a joint goal that needs to be solved and is discussed in the next section.

2.5.2.2.3 Teamwork models

Teams exist out of a (temporary) collection of agents that have interdependencies. These interdependencies occur when local decisions of agents influence decisions of other agents in the team or there is a possibility of conflicting interests between team members. In Distributed AI, problem solving agents work together in order to solve their local goals and the goals of the community as a whole (Jennings, 1995). Each individual agent can be assigned its private beliefs, desires and intentions and can communicate with other agents. Assume a purely self-interested agent that acts in isolation and tries to optimise its own local performance with no concern about the overall outcome of the process, e.g. a sales-agent sells and promises more than a production agent can ever produce. To prevent such situations, there is a need for coordination. Coordination is necessary when there are dependencies between agents' actions, when there is a need to meet global constraints and when an individual does not have sufficient competence, resources or information to solve the entire problem (Jennings, 1996).

Cohen and Levesque (1991) discuss teamwork and coordination, i.e. what is involved in doing something together. They extend the model of the agent with a belief-goal-commitment model of the mental states of individuals (Cohen & Levesque, 1990)). The extended model specifies intentions as *internal commitments* to perform an action. A team is considered as a collection of agents to which the mental state of joint intention can be assigned: a joint intention is a joint commitment to perform a collective action while in a certain shared mental state, as the glue that binds team members together (Cohen & Levesque, 1991). The formation of a joint commitment is established when all members have mutual belief of a joint persistent goal[25].

Next, being part of a team implies a certain responsibility towards other members, i.e. when one of the members drops the individual commitment—it thinks the goal is finished, or the goal will never be satisfied, or the motivation for the goal is no longer present—then the agent has the persistent goal to notify other agents of this fact. Jennings (1995) states that not only agents must have a

[25]The model of Cohen and Levesque is at the rational level, which assumes that the collection of actors has a joint intention and does not take into account the cognitive mechanisms at lower levels. In the next chapter, we will approach the organisation as being a social construct and as a set of agents that have each their own representation of the organisation.

joint goal and the wish to achieve that goal, but also a common recipe for attaining the joint goal, see figure 2.7. The difference is that when the commitment to the goal is dropped, the joint action is over. However, when the commitment to the recipe of actions is dropped, the team still has the ability to come up with a new recipe of actions that could lead towards the joint goal. The *Joint Responsibility Model* is introduced as an explicit model that requires: "[...] all the team members to have a joint persistent goal to attain their joint goal X, that they all execute the common recipe Y according to the principles of joint recipe commitment, and moreover, that all the agents mutually know what they are doing whilst they are doing it" (Jennings, 1995, p. 209).

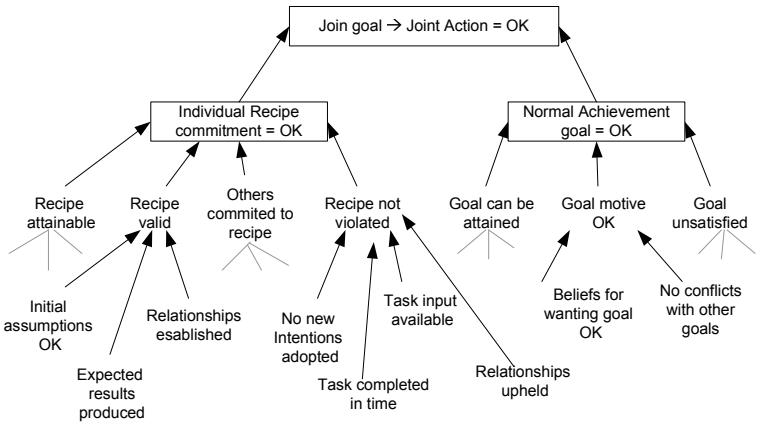


Figure 2.7: Causal network for commitment to joint goal (adapted from Jennings, 1995).

The Joint Responsibility Model is implemented in an architecture, the ARCHON (Architecture for Cooperative Heterogeneous ON-line systems) framework (Jennings & Pople, 1993; Wittig, 1992). This model is applied in industrial control systems such as GRATE* (Jennings, 1995), see figure 2.8.

The ARCHON architecture is a high-level architecture and functions as a cooperation module that can be placed on top of (in software terms: wraps) software legacy systems. The architecture distinguishes two separate layers, the *cooperation and control layer* and the *domain level system*. Within the cooperation layer, there are three main problem solving modules.

> The *control module* is the interface to the domain level system and is responsible for managing all interactions with it. The *situation assessment module* makes decisions which affect both of the other two modules. It decides which activities should be performed locally and which should be delegated, which requests for cooperation should be honoured, how requests should be realised, what actions should be taken as a result of freshly arriving information, and so on... The cooperation module is responsible for managing the agent's social activities, the need being detected by the situation assessment module. Three primary objectives related to the agent's social role are

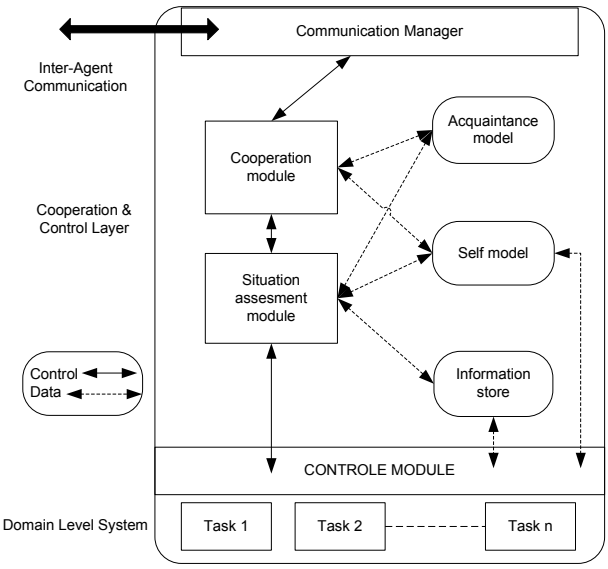


Figure 2.8: ARCHON (GRATE*) agent architecture (Jennings, 1995).

> supported. Firstly, new social interactions have to be established (e.g. find an agent capable of supplying a desired piece of information). Secondly, ongoing cooperative activity must be tracked (using the Responsibility convention). Finally, cooperative initiations from other agents must be responded to. (Jennings, 1995, p. 212)

The situation assessment module retrieves situation-dependent information from (1) the *acquaintance model*—skills, interests, currents status of others, (2) the *self model*—skills, interest of the agent self, (3) the information store that keeps up the results of the underlying domain level system, and (4) the results of interaction with others. The acquaintance model is a (mutual) modelling approach (see next section) that enables an actor to anticipate on actions of other agents and achieve quicker and better performance in coordination problems.

2.5.2.2.4 Coordination by mutual modelling[26]

Agents that have to survive in an environment must be equipped with perception, action and memory components that give the ability to learn and anticipate on new situations. Mutual modelling creates explicit models of other agents in the memory of the agent. For example, a kid constructs an intentional model of its parents in memory. Based on experience, it knows that its mother is more giving in to complaints than dad. So whenever it needs consolation or something to be done, the kid will pester its mum. On the other hand, dad has

[26]The reader probably notices that we are shifting from a top-down approach (DPS) towards the bottom-up approach (MAS). Although there is a distinction between both, they are often interweaved with each other.

a model of (the interaction of) the kid and his wife and knows that in order to prevent mum from giving in too easily, he will enforce his wife to be more strict with their kid. The father puts himself in the position of his wife allowing him to (indirectly) coordinate the actions of the child more effectively, i.e. he has an internal model of his wife—the beliefs, desires, intentions, and so on—to be able to cooperate and create predictions about the way his wife and his child behaves. Mutual modelling creates a better mutual understanding between agents. Jennings (1992) states that mutual modelling plays an important role in:

- *Focusing Activity and Reducing Communication*
  Agents with knowledge about other agents' intentions and capabilities are more effective and efficient in coordinating their actions with other agents. The amount of knowledge gathered about other agents creates the possibility to predict how other agents behave. The gathered information reduces the need for communication between agents and gives better directions to select the right actions.

- *Coordination and Coherence*
  Mutual modelling allows agents to maintain models of each other and prevent disturbing each other goals while attempting to achieve the joint goal, i.e. it lowers the probability of conflicts and improves coordination between agents. It also increases coherence or effectiveness with which the primitive actions in a complex action conspire to bring about a particular joint goal (Wooldridge, 1994).

- *Self Reasoning*
  When the agent is able to compare local capabilities with capabilities of others it can reason what tasks can be solved by himself and what tasks require social interaction, what information must be available to initiate a task, what results will be produced and when a particular activity will be finished.

The first application that started to test mutual modelling and MAS was MACE (Gasser, Bragamza, & Herman, 1987). MACE constructs a database of acquaintances and adapts the knowledge based on changes in the environment. The agent has a model in which it stores:

- Class: groups of agents are identified with a class name.
- Identification of the modelled agent by name, connected to a class.
- Roles: the function the modelled agent has in that class.
- Skills: skills that are connected to that role.
- Goals: the goals that the modelled agent wants to achieve.
- Plans: the means of the modelled agent to achieve the goal.

Whereas some of the knowledge stored of other agents concerns physical characteristics, much of the knowledge is referring to socially constructed artefacts created by human beings themselves. Social psychologists, such as Mead (1934) and Blumer (1969) (see chapter 3) indeed state that creating a mental representation of others is necessary in interaction with others. The coordination by norms and laws is closely linked to mutual modelling.

2.5.2.2.5 Coordination by norms and social laws

Coordination by norms and social laws is possible when agents can imitate and maintain models of other agents. Wooldridge (2002) states that a norm simply is an established, expected pattern of behaviour and that social laws exist out of norms to which authority and roles are assigned. Conventions or social constructs (see chapter 3) provide stable patterns of behaviour in otherwise unstable and uncertain situations. When applied in an agent society, two approaches can be distinguished. The first approach is the implementation of a fixed normative system that is designed and installed in the agent *off-line* (Shoham & Tennenholtz, 1995). The second approach (Shoham & Tennenholtz, 1992b) is the emergence of a convention between agents. Experiences with and about others are remembered and with help of a strategy or utility-function, the agent determines what type of behaviour will probably be most successful. Such behaviour becomes a habit of action when it is reinforced by all agents (mutual reinforcement) that perceive the behaviour as being successful. Mutual reinforcement of behaviour can eventually lead to the emergence of norms that prescribe what behaviour is desired in certain situations. In the establishment or spreading of norms, there can be made a distinction between reactivation of the same representations under the effects of a given perception (social-facilitation) and behaviour spread by social cognitive processes mediated by the agents' social goals and beliefs (imitation), i.e. exchange of social constructs (Conte & Paolucci, 2001).

In chapter 6, based on a traffic law system (cf. Kittock, 1993; Epstein, 2001; Shoham & Tennenholtz, 1995), two experiments demonstrate the spreading of norms. The first experiment demonstrates social facilitation or implicit change of cognitive preferences for certain rules, influenced by perception and reward. The second experiment demonstrates the explicit transfer of the mental representation of a norm by an authoritative agent to an other agent. In the next chapter, we will explain social constructs that serve as coordination mechanisms and hold knowledge about mental attitudes of others and social structures (e.g. authority and roles).

Although the field of MAS provides much more as can be shown in this chapter, we will end this section. Still, the reader is entitled to have some insights into applications that have emerged, especially with the growth and importance of the Internet, the first generation of applications has entered the market.

2.6 Multi-Agent System Applications

A large variety of Multi-Agent Systems are available on the commercial market and as open-source on the Internet. Agent technologies are not restricted to academia, companies and research labs anymore, but they have entered gradually into our computer systems and homes. In solving problems that concern interactions between people, the application of Multi-Agent Systems has proven that it is a good candidate, i.e.

> Multi-agent systems are ideally suited to representing problems that have multiple problem solving methods, multiple perspectives and/or multiple problem solving entities. Such systems have the traditional advantages of distributed and concurrent problem solving, but have the additional advantage of sophisticated patterns of interactions. Examples of common types of interactions include: cooperation (working together towards a common aim); coordination (organising problem solving activity so that harmful interactions are avoided or beneficial interactions are exploited); and negotiation (coming to an agreement which is acceptable to all the parties involved). It is the flexibility and high-level nature of these interactions which distinguishes multi-agent systems from other forms of software and which provides the underlying power of the paradigm. (Jennings et al., 1998, p. 9)

First, as mentioned before, Multi-Agent Systems consist of agents that have incomplete information, no centralised control, data is decentralised and computation is asynchronous. Secondly, the weak notion of agency states that agents are flexible and in general have the properties of autonomy, social ability, reactivity and pro-activeness. And thirdly, we have distinguished different types of agents, i.e. the cognitive and the social agent. Creating a classification of Multi-Agent Systems is a tedious task and we could take an approach that is based on the types of agents applied and their interaction with the environment resulting in a classification of reactive towards social pro-active systems. However, the difference in opinions and definitions about what agents are, automatically results in a problematic classification of MASs.

An approach by Wooldridge (2002) is to first draw the distinction between distributed systems—multiple agents are nodes in a distributed system—and agents that have the (pro-active) function of assisting users working with applications (Maes, 1994). Next, Luck, McBurney, and Preist (2004) divide MAS into (real-time) Multi-Agent decision systems—agents participate in a system have to make joint decisions, and Multi-Agent simulation systems, where MAS is applied as a model to simulate some real-world domain. The important distinction between both is that with decision systems, there is often a direct outcome ready for application in the real world, while simulation systems require first an interpretation of the researcher before results can be applied in the real world[27].

[27] We are aware of the fact that games or entertainment software are somehow in the middle, but most of the time, games have no direct impact on the world (e.g. someone shot in a movie does not create a real murder case).

Nevertheless, we follow the approach of Luck et al. by classifying systems in two groups and show in which domains MAS applications are developed. The domains can be structured varying from hardware, such as robotics and manufacturing towards software, e.g. simulation systems and entertainment. In the next sections, 2.6.1 and 2.6.2, we respectively mention examples of multi-agent decision systems and multi-agent simulation systems.

2.6.1 Multi-Agent decision systems

The *Industrial Application* is one of the domains where MAS needs to interact with the physical world, be it humans or other agents. Industrial manufacturing and (real-time) control systems in the form of MASs are introduced to solve coordination problems in factories, air-traffic control, telecommunication networks and transportation systems. The general goal of industrial applications is to manage processes efficiently. Examples of industrial applications are the manufacturing system YAMS (Parunak, 1995) and the process control system ARCHON (Jennings & Pople, 1993). Closely linked to manufacturing systems (primary processes) are applications for workflow and business process management (secondary processes). An example is the ADEPT project that views the business process "[]...as a collection of autonomous problem solving entities that negotiate with one another and come to mutually acceptable agreements that coordinate their interdependent sub-activities" (Jennings, Faratin, Norman, O'Brien, & Odgers, 2000, p. 2).

Industrial applications are mostly custom-made solutions for specific problem areas in the Business to Business market, while *commercial applications* are more directed towards the consumer market. Coordination of interdependent sub-activities, similar to ADEPT, can also be applied in information retrieval and sharing systems. A 'web' of agents that are experts in a certain area of interest form a connected network of information sharing nodes. Other agents function as brokers and have knowledge about where to go for what. In P2P applications, e.g. Gnutella, Bit Torrent and Skype, this functionality is already delivering music, video and voice over IP. Following up the information agent network, e-commerce is the next step in which (mobile) agents take over tasks from the user and 'walk' over the Internet to compare prices and find the most suitable deals with supplier agents. For instance, Chavez and Maes (1996), define a web-based multi-agent auction system (Kasbah) in which humans are represented by bidder and seller agents.

MAS also entered the entertainment and leisure industry, where agents play a part in computer games, e.g. the Creatures (Grand & Cliff, 1997), movies (Lord of the Rings, Massive Limited[28]) and virtual reality applications. Commercial applications of Multi-Agent decision systems probably will surround us in the near future and offer services that today can be spotted in Science Fiction movies, e.g. a car that is equipped with a diagnosis agent will automatically offer its service to the driver when the car needs maintenance; it can arrange an appointment with the garage that fits the schedule of the driver. Such a service

[28] http://www.massivesoftware.com

is established by multi-agent interaction between many agents in order to get the best quality combined with the lowest possible price.

The entertainment industry and application service providers have created a surrounding in which the physical world and the virtual world become more and more connected. We see the connection between the virtual and physical world also in the area of multi-agent simulation systems.

2.6.2 Multi-Agent simulation systems

Agent based simulation systems are used to describe, prescribe and predict real-world scenarios and are applied in many areas, e.g. traffic systems, crowd behaviour and riot control, combat scenarios and many alike. In this dissertation, the (cognitive) agent-based social simulation (Gilbert & Troitzsch, 1999; Sun, 2006a) is applied as a modelling tool for studying the behaviour of agents that represent individual people. In the field of agent-based social simulation, two broad approaches can be distinguished (Luck et al., 2004). One approach—the emphasis in this dissertation—defines systems that underlie social interaction, and the other focuses on observing social processes and modelling those. The combination of both approaches can be an iterative process, by first analysing the field, next creating a multi-agent simulation system and finally observing and validating the model.

An example of an application is given by Wooldridge (2002) who mentions a social system, the EOS project (Doran & Palmer, 1995) that consists of agents making it possible to describe a number of social phenomena, such as 'overcrowding'—when too many agents attempt to obtain resources in some locale or 'clobbering'—when agents accidentally interfere with each other's goals. Another well known example of a simulation system is Swarm (Minar, Burkhart, Langton, & Askenazi, 1996). Swarm is a generic discrete event model used for simulating a collection of independent agents of any kind. The areas in which Swarm already is applied is diverse, e.g. chemistry, economics, physics, anthropology, ecology, and political science.

Many simulation toolkits have evolved and the search for adopting a toolkit that is suitable for our current research has been difficult. Therefore, in the discussion of this chapter, we explain why we invest in constructing a new toolkit.

2.7 Discussion

The aim of this dissertation is the creation of multi-agent based social simulations[29] whose agents are grounded with help of a plausible cognitive architecture and are equipped with the capability of social construction[30] in order to express social or normative behaviour.

In this discussion, first we want to explain the type of agent that is suitable for our research purpose and in the second place, we want to discuss what agent

[29] Chapter 1, research question 1.1.

[30] Chapter 1, research question 1.2; see also next chapter.

toolkit would be appropriate for modelling a cognitive agent-based computational social simulation model that can explain interactive social behaviour.

In order for an agent to exhibit intelligent social behaviour, the agent should be equipped with a cognitive architecture[31], i.e. a physical symbol system that makes it cognitive plausible and allows it to exhibit intelligent action. Apart from that, the agent should also be able to socially construct its world and show social or normative behaviour, i.e. it should be physically and socially situated. The physical and social situatedness requires the capabilities of a reactive agent (embodied/situated cognition) that responds to changes in its environment and of a social agent that takes needs of other agents into account. Therefore, the type of agent applied in this dissertation should be a hybrid agent, existing out of the capabilities of a cognitive, a reactive and a social agent.

The first step in this research was to find a suitable cognitive agent or a suitable agent toolkit. This requirement restricted our options dramatically, because the amount of plausible cognitive architectures is low. The most well known cognitive architectures are SOAR (Newell, 1990) and ACT-R (Anderson & Lebiere, 1998). In chapter 4, we will explain what type of architecture we have adopted for constructing a cognitive plausible agent.

The next step was to see how to create physical and social situatedness in our cognitive actor. As discussed in section 2.4, three types of environments need to be supported by our simulation toolkit, i.e. the physical, communication and social environment. The hybrid agent that responds to physical and social changes in a simulated environment, requires (the development of) a simulated physical, communication and social environment that allows for the exchange of social constructs and norms. In the last step we added a social or normative layer to the cognitive agent (cf. ARCHON: Wittig, 1992) and applied subsumption (Brooks, 1986) that allows the agent to beware of changes in the social environment. These three steps resulted in a cognitive agent-based social simulation. The model and design of the complete (social and cognitive) agent, i.e. RBot (cf. Roest, 2004) and the simulated environment (Multi-RBot System/MRS) will be discussed in chapter 5.

In chapter 3 and chapter 4, we will study the individual agent, respectively the social and the cognitive actor, and will discuss what is required for an actor to exhibit *cognitive plausible social behaviour*. Chapter 5 will discuss the implementation of these requirements in a multi-agent model—RBot and MRS—that will provide us with a multi-agent platform for conducting demonstrative experiments in chapter 6.

[31] Chapter 1, research question 1.3; see also chapter 4.

CHAPTER 3

The Social Actor

THE previous chapter has explained the field of Multi-Agent Systems. In the discussion about Multi-Agent Systems it has become clear that we need a hybrid agent, one that is cognitive, reactive and social, i.e. the cognitive agent has to be physically as well as socially situated in the environment. In the next chapter, we explain what is understood by a cognitive actor and a reactive actor. In this chapter we want to focus on the social level of agency, i.e. the social aspects of an agent, and how the social agent is embedded in the social environment.

The first question that comes to mind is: why is it necessary to study and model a social actor, and what reasons are driving an actor to behave socially? The number of reasons to behave socially is plenty. One of them could be that social behaviour can reduce uncertainty. For instance, the uncertainty reduction theory of Berger and Calabrese (1975) assumes that: "...[] when strangers meet, their primary concern is one of uncertainty reduction or increasing predictability about the behaviour of both themselves and others in the interaction" (p. 100). Hence, when reduction of uncertainty is an intrinsic desire of humans, it requires communication, social interaction and exchange of social attitudes in order to create certain predictions about behaviour of others. Another possible reason for social behaviour is reciprocity (Fehr & Gächter, 2000) or altruism. Altruism could be seen as the reason why self-interested economic utility calculations mostly do not work, i.e. the question is that: "...[] when we help others, [is it] because our ultimate goal is to benefit them, or whether our ultimate goal is always some more-or-less subtle form of self-benefit" (Batson & Shaw, 1991, p. 159). There are many other reasons and apparently there is an indication that there is an intangible aspect in behaviour that can be addressed as a form of social behaviour that occurs between individuals and simply cannot be understood by economic utility calculations alone.

This chapter focuses on social aspects of the actor that enables it to behave socially. These aspects should create the possibility for actors to express social behaviour and become aware that they are not alone, but are also part of a group or society. In other words, the actor does not only live in his own world (internal representation) but also builds up relations with the outside world; with physical, social and cultural objects, actors and groups of actors.

Sociology and social psychology are research fields that focus on interaction between (processes of) individuals and groups. In our research we take social constructivism as a theory that focuses on the understanding of context and social interaction, and the assumption that knowledge creates and is created out of social processes. The second theory we adopt is semiotics that explains how knowledge and its meaning are created and transferred between individuals with the help of signs. And the third theory is cognitive science/psychology, which is discussed in chapter 4.

Social constructivism as a social psychological theory attempts to explain the relation between the individual as part of its society. In our line of reasoning, the theory follows the right direction, however its description is too abstract to transform into a useful social (actor) model that is (directly) applicable for our use in a simulation setting. Therefore, we adopted levels of description as shown in figure 3.1 and discussed in chapter 1. The *intentional level* and *functional level* are concerned with the individual's mind (respectively its intention-and-belief system and the learning/cognitive system); we elaborate more about these levels in the next chapter. The *social level* is concerned with behaviour of groups or being/feeling part of a group; it addresses topics like knowledge of the generalised other, habits of action, social laws, collective action and the overall characteristics of the group. The *semiotic level* describes the use of language, codes and signs existing in a community and used for communication, interaction and negotiation. With help of a signification system and sign production, these signs can be understood and produced by actors. It seems that the semiotic level is a level that is present (as signs) in the community and in the individual as well.

In section 3.4, we will elaborate more about the semiotic level and its connection with the social and individual (intentional and functional) level. Figure 3.1 also shows that the theories (cognitive/social psychology, sociology and semiotics) position or identify themselves with (pay most of their attention to) one of these levels.

We argue that in order to construct a model of the social (and cognitive) actor, we need to understand all these levels of description and corresponding theories. These theories and the social actor will be explained in this chapter except for cognitive science that will be elaborated in the next chapter.

The outline of this chapter is structured in the following way (see also figure 3.1). In section 3.1, we introduce the chapter with social constructivism and argue that an actor socially constructs its world, i.e. the actor builds up representations in its mind about the social and physical world. Section 3.2 argues that an organisation exists by means of a collective social construction and that an organisation can be traced back to representations in the mind of the actors or members of the organisation. In section 3.3, we introduce the social environment

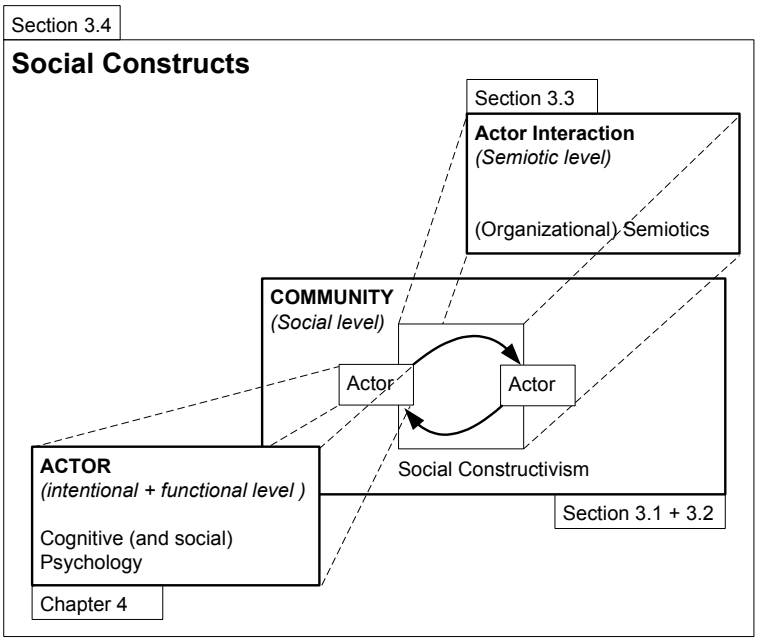


Figure 3.1: Social constructivism and overview of levels and theories.

and introduce the theory of semiotics and communication. We argue that an actor in order to behave socially not only requires the support of a communication process and its communicative functions, but also that such a communication process presupposes a signification system and sign production based on codes. In section 3.4, we start the discussion about the social actor[1] and introduce the generic theoretical framework of social constructs. Social constructs are representations that exist in the mind of the actor, and physically as artefacts in the external world (documents) and can function as coordination mechanisms for the actor in order to exhibit social behaviour. Finally, in section 3.5, we close the chapter with a discussion about the requirements for an actor to exhibit social stable behaviour and argue that representations in the mind of the actor need support from a cognitive architecture; an argument that is more elaborated in the next chapter. In chapter 5, these requirements serve as guidelines for the design and modelling of the cognitive and social actor RBot.

3.1 Social constructivism

The founding principles of social constructivism were established by Mead (1934) and Vygotsky (1962, 1978). Mead analysed the human being as a complex individual who socially constructs the world, itself and other human individuals. The social construction of the world is created by a process of interaction, but what is social construction and what can and cannot be socially constructed?

[1]The cognitive actor, as shown in figure 3.1, will be discussed in chapter 4

A concrete definition of what social construction is, seems impossible because the term social construction has become blurred by its popularity and overuse in literature for anything that seems worthy constructing (Hacking, 1999). Hacking found out that many authors use a localised view on social construction to raise consciousness for addressing certain problems in our world, e.g. women refugees, homeless people etc. Hacking states that the following will hold for social construction in general:

> X[, as a social construct,] need not have existed, or need not be at all as it is. X, or X as it is at present, is not determined by the nature of things; it is not inevitable. . . . X was brought into existence or shaped by social events, forces, history, all of which could well have been different. (Hacking, 1999, pp. 6–7)

Hence, a social construction or social construct can be seen as an invention or artefact (cf. Goodman & Elgin, 1988; Simon, 1996) constructed by interaction between members of a social group or interaction between groups. Products of social construction, such as institutions, gender and emotions are "...social constructs, created, disseminated, and agreed upon by social groups..." (Sismondo, 1993, p. 522).

The social constructivist Blumer (1969), a student and follower of Mead, states that Mead started the reasoning about social construction by developing a reversed way of thinking stating that not only human's actions create the (social) world but that also the social world creates the mind and consciousness of the individual human being:

> His [Mead's] treatment took the form of showing that human group life was the essential condition for the emergence of consciousness, the mind, a world of objects, human beings as organisms possessing selves, and human conduct in the form of constructed acts. He reversed the traditional assumptions underlying philosophical, psychological, and sociological thought to the effect that human beings possess minds and consciousness as original "givens", that they live in worlds of pre-existing and self-constituted objects, that their behaviour consists of responses to such objects, and that group life consists of the association of such reacting human organisms. (Blumer, 1969, p. 61)

Hence, the assumption is that there is an interdependent relationship between human group life (the society) and the human being itself, because without social interaction with other human beings, we do not get acknowledgement of our existence, our mind and ourselves, i.e. without the others and the (social) act we are unable to construct a reality/society for ourselves. Blumer, as well as Mead base their social psychological theory on concepts like the act, the objects, the social act and society that address how human society is constructed by interaction of individuals. We will discuss these concepts below.

The act Human action is not merely a product of factors that play upon or through the human actor. In social constructivism, the act is constructed by the actor and therefore action will not be formed by simply stating that the response is solely dependent on the conditions of stimuli. Human action and the individual can be represented by the play and the game. The play is often found in children that have for instance invisible imaginary friends, in which the child responds to its own stimuli and becomes aware of itself. On the other hand, in the game, the child must perceive the attitudes of the others that define a relationship between itself and the rest. The organisation of the game is arranged with the help of (in) formal (socially constructed) rules. The game creates the possibility to take into consideration the particular attitudes of other individuals with whom the individual interacts; thereby creating social acts with each of these individuals. After this stage of social interaction, it is possible to organise the others into a sort of unit or community that can be designated as 'the generalized other':

> The organized community or social group which gives to the individual his unity of self may be called 'the generalized other'. The attitude of the generalized other is the attitude of the whole community. (Mead, 1934, p. 154)

An example can be seen in how a member of a political party takes on the generalised attitude of that party and consequently speaks or responds in terms of the organised attitudes of that party, i.e. the member feels itself committed to a community and represents thereby that community.

Objects An extreme form of social constructivism assumes that all objects are human constructs and that they do not exist as self-existing entities with intrinsic natures. Hence, any object is a social construct, be it a company, a marriage-contract, an imaginary friend or a rock as long as they are assigned a meaning[2], or are interpreted by human beings and are therefore part of our world.

According to Searle (1995), the world view depends on our concept of objectivity and the contrast between the 'objective' and the 'subjective'. He proposed two senses of the objective-subjective distinction: the epistemological and ontological objective-subjective distinction. The epistemological distinction is applicable to judgements, in which a judgement is *epistemologically objective* when its truth or falsity is independent of attitudes or feelings of people, and *epistemologically subjective* when it is based on certain attitudes or feelings of people.

[2]Meaning arises and lies within the field of the relation between the gesture of a given human organism and the subsequent behavior of this organism as indicated to another human organism by that gesture. If that gesture does so indicate to another organism the subsequent (or resultant) behavior of the given organism, then it has meaning. In other words, the relationship between a given stimulus—as a gesture—and the later phases of the social act of which it is an early (if not the initial) phase constitutes the field within which meaning originates and exists. Meaning is thus a development of something objectively there as a relation between certain phases of the social act; it is not a psychical addition to that act and it is not an "idea" as traditionally conceived. (Mead, 1934, pp. 75–76)

The following two statements: (1) "The mount Everest is an ugly mountain" and (2) "The mount Everest is the highest mountain on earth" are examples of epistemological subjectivity and objectivity. The former is a subjective notion of what is beautiful or not and the latter is an objective fact of the 'real' world independent of attitudes or feelings regarding its height. The *ontological* sense of objective and subjective is applicable, not to statements or judgements, but to entities, objects or properties. The moon, for instance, is an example of an *ontological objective* entity; with or without the presence of human beings, the moon will inevitably be there. On the other hand, pain is *ontologically subjective*, because its existence depends on being felt by the subjects itself. The subjective object is always dependent upon the meaning it has for a person or person(s). Such a meaning is not part of the object, but socially constructed (formed and transformed) by the person(s) involved. People are prepared or set to act based on the meaning of an object, e.g. a kitchen knife can be a tool for a cook, but a weapon for a murderer[3]. Searle refers to the (meaning of a) knife as being an observer-relative object, because it exists relative to the intentionality of the observers and users. On the other hand, the moon for instance is an *intrinsic* object, its existence is independent of the observer and intrinsic to nature, i.e. the mass of or chemical composition is independent of the observer. Simon (1996) refers in a similar way to objects; he states that an object or artefact functions as an interface "between an 'inner' environment, the substance and organization of the artefact itself, and an outer environment, the surroundings in which it operates" (p. 8). An individual not only interacts with objects or artefacts; an individual also interacts with humans (and animals) who are assumed to have an own will and in contrast to objects, the interaction with individuals is of a social nature.

Social act and society The social act is described by Blumer as a "collective *form* of action that is constituted by the fitting together of the lines of behaviour of the separate participants" (emphasis added: Blumer, 1969, p. 70). In the individual act, the human being depends on its own interpretation of the interaction, whereas in the social act, the human being has to bring its behaviour in line[4] with the others, e.g. members of a removal company have to align their acts for not breaking anything. The type of interaction can vary from bilateral acts, a marriage, and a trade-contract, towards multilateral acts, e.g. an opera, a company activity, or a government of a country. Because the social act is a collective form that depends on individual actions of each separate individual and each individual develops its own experience over time, the social act is exposed to many uncertainties. The start of the social act can be complicated and when it has commenced, it can be interrupted, abandoned, or transformed. When individuals have given different meanings to what the social act stands for, there is a possibility that they do not act in line with each other, which causes actors to deviate from each other. Hence, social construction can only *support* individuals

[3]According to the affordance theory (Gibson, 1979), we perceive affordance properties from the knife, i.e. we perceive possibilities for actions: the cook cuts the vegetables and the murderer kills its victim.

[4]We assume that when *social* actions need to be in line with others, it requires a form of coordination (mechanism)

in forming a social act; the diversity of experience and understanding can always give rise to difficulties in a common understanding or meaning. A human society is made up of individuals who have selves or better, are entities with cognitive capabilities for reasoning about the world that surrounds them, i.e. individual action is a construction and not merely a release of stimulus-response behaviour. Individuals accomplish social acts that "...[] consist of the aligning of individual actions, brought about by the individuals' interpreting or taking into account each other's actions"(Blumer, 1969, p. 82). The process of alignment of individual behaviour towards a coherent act can be seen as the start of the social construction of a formal organisation or institution. However, in order for an organisation (or institution) to exist in a society and formally be accepted, it needs to be explicitly constructed in a negotiation process. After the establishment of an organisation, it can function as a unit for storing documents that serve as guidelines for performing social acts. In the following section, we discuss the organisation and society as carrier and provider of products of social construction.

3.2 Social construction of organisation and society

In the previous section, we have stated that the social act is a mutual effort for coordinating activities. Such a social act or a set of interdependent social acts, observed as a coordinated activity, constitutes as a community, or according to Mead (1934):

> [A community]... is dependent upon individuals taking the attitude of the other individuals. The development of this process... is dependent upon getting the attitude of the group as distinct from that of a separate individual-getting what I have termed a "generalized other". (p. 256)

Hence, the moment we detect a 'generalized other', we discover a community that conforms to certain ways of conduct or habits of action that exist in that community. A community can be an informal community or a formally acknowledged community, which we can refer to as an organisation. Such a community or organisation delivers a common response, from society to individuals of that society, or as Mead (1934) states:

> There are, then, whole series of such common responses in the community in which we live, and such responses are what we term 'institutions'. The institution[5] represents a common response on the part of all members of the community to a particular situation. (p. 261)

Mead does not discuss the *cognitive* aspects of individuals, and we are aware that today "social cognition has become the dominant perspective in social psychology (Schneider, 1991)" (Tenbrunsel, Galvin, Neale, & Bazerman, 1996, p.

[5]Later on, see section 3.4.3, we give a better description of institution, i.e. according to Rawls (1971) an (explicit and formal) institution is a public system of rules, which defines offices and positions with their rights and duties, powers and immunities.

315). Social cognition studies interactions between people, and how people make sense of others, but in social cognition, in contrast to social constructivism, the emphasis is on how cognitive processes influence social behaviour (Fiske & Taylor, 1991). In this dissertation, see chapter 4, we apply this cognitive approach in studying social interaction between individuals, i.e. we adopt theories of cognition—a cognitive architecture—to model and study how cognitive plausible actors influence the behaviour of others.

Taking into account cognition, we can state that the organisation exists as a formally acknowledged entity that is maintained by interaction between individuals—its members. But does this statement claim that an organisation is a tangible (intrinsic of nature) object or/and that an organisation is an intangible (observer-relative) object, i.e. does an organisation'really' exist? Recall that for an object to be ontological objective it should still be existent in the absence of minds and to be ontological subjective it should depend on minds, and to be epistemological objective, the judgement about facts should be independent from attitudes or feelings about these facts. We give the following example of the five dollar bill to show how an organisation should be thought of as an object.

> For example, for some kind of physical object to be money, we as society must think of it as money. These attitudes are partly constitutive of its being money. The claim that I have $5 in my pocket is epistemologically objective. It is made true by the money being there, whether or not any particular person knows about it. On the other hand, there is a sense in which this fact only exists as a result of collective human acceptance. That is, whereas it holds independent of beliefs or attitudes, it is not independent of all human cognition. It is thereby ontologically subjective. (Meckler & Baillie, 2003, p. 281)

In this example, when we substitute money with organisation, it becomes clear that an organisation is epistemologically objective and ontologically subjective, except that I do not have the organisation in my pocket, or do I?

An organisation is an entity to which people collectively have assigned a function to, i.e. the tax office its function is to collect money from people for the public good, and the coal mine has the function to deliver coal to industry and households as a source of energy. An organisation exists merely because we collectively agree it to be an organisation, i.e. the organisation does not exist in the absence of such an agreement (Meckler & Baillie, 2003). Therefore, it is in this sense that the organisation is socially constructed and ontologically subjective.

In this dissertation, we adopt this constructivist point of view in which the organisation exists as a social construction created, observed and acknowledged by individuals. The reason is that coordinated activities such as organisations in MAS are explained by the assumption that organisations exist based on social interaction between autonomous agents. Further, we assume that an organisation can be traced back to representations (in the mind of actors) that structure the interaction among people, thereby demarcating the group of people that are members of the organisation (Van Heusden & Jorna, 2001).

In the research conducted by us, we state that the organisation exists out of several actors, producing and perceiving symbols and/or signs to interact with each other. The organisation is a constructed reality as long as the interacting actors and their teamwork exist (Gazendam & Jorna, 1998), i.e. organisations exist as long as there are representations in the mind of the ones—insiders or outsiders—who perceive(d)[6] the organisation. Social constructivism creates the awareness that organisations exist out of individuals who together construct that same organisation by symbolic interaction. Social constructivism, concerned with symbolic interaction and meaning, is closely linked with semiotics. The discipline of semiotics concerns signs[7], and describes how the meaning of signs depends on—and is transferred during—the social interaction between individuals.

3.3 Social Environment and Semiotics

What brings together social constructivism and semiotics is the *meaning* of an 'object'. While social constructivism is focused on the (social and cultural) construction of the meaning of the object for the individual, *semiotics* is concerned with the *creation*, *use* and *transfer or communication* of the meaning with the help of *signs*, i.e. semiotics studies all cultural processes as processes of communication and signification (Eco, 1976).

Semiotics is concerned with the construction of signs in the environment, a signification process through which certain systems of signs are established by virtue of social and cultural conventions or constructs. Conventions or constructs such as books, laws and contracts can act as a memory aid for enhancing (situated) cognition of individual and organisation.

The transfer and the construction of meaning—in which the individual is a sign producer as well as an interpreter—are influenced by the environment of the individual that puts constraints on the processing of available signs. Besides that, the individual is limited in the range of signs it can perceive, determined by the biological make-up of the individual itself.

3.3.1 Umwelt

The individual determines its behaviour patterns based on the environment or semiotic *Umwelt* in which it is situated. Such a semiotic Umwelt is described by Von Uexküll as follows:

[6] For example, when an organisation bankrupts, it does not formally exist anymore. However, its informal existence depends on the moment that all actors who (have acted based on the formal existence of the organisation and perceived that organisation) have no beliefs or memory about its existence anymore.

[7] *Semiotics is concerned with everything that can be taken as a sign.* A sign is everything which can be taken as standing for something else. This something else does not necessarily have to exist or to actually be somewhere at the moment in which the sign stands in for it. Thus *semiotics is in principle the discipline studying everything which can be used in order to lie.* If something cannot be used to tell a lie, conversely it cannot be used to tell the truth: it cannot in fact be used 'to tell' at all. (Eco, 1976, p. 7)

> []...first blow, in fancy, a soap bubble around each creature to represent its own world, filled with the perceptions which it alone knows. When we ourselves then step into one of these bubbles, the familiar meadow is transformed. Many of its colourful features disappear; others no longer belong together but appear in new relationships. A new world comes into being. Through the bubble we see the world of the burrowing worm, of the butterfly, or of the field mouse; the world as it appears to the animals themselves, not as it appears to us. This we may call the phenomenal world or the self-world of the animal. (Von Uexküll, 1957, p. 5)

Von Uexküll (1957) gave an example of the tick to illustrate the concept of *Umwelt* and how the individual is embedded in its world through *functional circles* (see figure 3.2). Sequential circles of interaction between the tick as a subject (as a *meaning-utilizer*) and a mammal as its object (and *meaning-carrier*) show the change of functional circles in its Umwelt (Ziemke & Sharkey, 2001).

- (circle 1) The tick awaits in the bushes for a mammal that passes by closely enough to receive perceptual cues (*Merkmal-Träger*) of butyric acid that are transformed into perceptual signs (*Merkzeichen*) which are processed by the perceptual organ (*Merk-Organ*). Next, the effector organ (*Wirk-Organ*) produces effector signs (*Wirkzeichen*) that creates effector cues (*Wirkmal-Träger*). This results into signs that are sent towards the feet in order for the tick to drop onto the mammal.
- (circle 2) The shock of landing terminates the first circle and serves at the same time as the perceptual cue that makes the tick move towards the skin.
- (circle 3) When the tick receives the heat of the skin of the mammal, the tick terminates the previous circle and the heat starts to become a perceptual sign in the tick's Umwelt. Finally, after producing the effector signs, the biting motions are produced.

Stamper (1973) developed the 'information field' independently of the concept of 'semiotic Umwelt', but the information field can be seen as the form that the semiotic Umwelt takes for a person living in a community. "Because each person generally lives in several communities (family, work, religious community, club, country, and so on), the semiotic Umwelt for a person is composed of all the information fields bound to the communities he or she participates in" (Gazendam & Liu, 2005, p. 7).

From a semiotics point of view, the functional circles describe a circular stimulus-response system that reacts on *signals*[8] or according to Sebeok (1994)

[8] A signal is a pertinent unit of a system that may be an expression system ordered to a content, but could also be a physical system without any semiotic purpose...A signal can be a stimulus that does not mean anything but causes or elicits something; however, when used as the recognised antecedent of a foreseen consequent it may be viewed as a sign, inasmuch as it stands for its consequent (as far as the sender is concerned). On the other hand a sign is always an element of an expression plane conventionally correlated to one (or several) elements of a content plane. (Eco, 1976, p. 48)

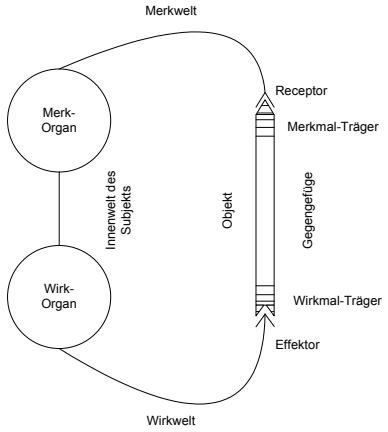


Figure 3.2: Functional circle (adapted from Von Uexküll, 1957).

this describes the Primary Modelling System (PMS). Besides that, Sebeok distinguishes also a Secondary Modelling System (SMS).

> All the animals paleontologists classify generically as Homo, and only such, embody, in addition to a primary modelling system. . . , a secondary modelling system, equivalent to a natural language. The difference amounts to this: while the Umwelten of other animals model solely a (for each) 'existent world', man can, by means of the secondary system, also model a potentially limitless variety of 'possible worlds' (containing sentences with alethic, deontic, or epistemic modalities). (Sebeok, 1996, p. 106)

The last and third modelling system Sebeok discerns is the Tertiary Modelling System (TMS): a model that describes abstract forms of semiosis that use conceptual symbolic representations (Gazendam, 2003). The complete system is a three-layered system: the TMS is grounded in language or speech (the SMS) and the SMS is grounded in the PMS.

The process of semiosis[9] and the unique ability of humans to consciously manipulate signs create the possibility for humans to construct worlds apart from nature and direct experience (Cunningham, 1998). The importance of signs in creating the Lebenswelt—the human Umwelt—lies in their creative power for infinite representation and meaning-making, or unlimited semiosis (Eco, 1990); a recursive process in which a sign refers to another sign and so on (Deely, 1990). Culture and society are therefore a collective construction, time after time reconstructed by the individual in interaction with others. The interaction with others is achieved by the production of signs, e.g. using sounds as a carrier, and is based on a process of communication. In the following section we discuss this interaction with help of processes of communication and semiotics.

[9]Semiosis is an activity, conduct or process by which an actor (part of a culture) produces signs and/or attributes meaning to signs, the process of meaning-making (cf. Chandler, 2002; Peirce, 1931).

3.3.2 Communication and Semiotics

Social construction requires the actor to interact and *communicate* socially with others in order to create representations in their mind that are also embedded in the social environment and shared with other actors. Communication is the practise of social relationships, and is central to the life of our culture, i.e. communication is social interaction through messages (Fiske, 1990).

Fiske states that there are two main schools of communication. The first school sees communication as *transmission of messages*; it is concerned with efficiency and accuracy of messages and sees communication as action or *communicative acts*. The second school is concerned with the *production and exchange of signs*, the discipline of semiotics and is considering *'works'* of communication, i.e. communication not only exist merely out of communicative actions, but the actor possesses a signification system (and therefore a theory of codes) and sign production as well (Eco, 1976).

The interpretation of social interaction between the two schools is different. The former sees influencing the behaviour of others as social interaction and focuses on the communication process, while the latter states that individuals involved in interactions are members of a culture or society and that messages are interpreted by the individuals based on their cultural experiences and shared understanding of these messages. Hence, the difference is that scholars of the first school do not interpret or evaluate the meaning of messages with reference to relevant codes in order to emphasise the *semiotic* (or social) codes involved, and thus do not highlight the social factors (Chandler, 2002, p. 175). In this dissertation, we adopt semiotics as a theory for communication[10], a more complete theory that gives attention to the relation between text, codes and culture, and which covers the aspects of communicative acts as well.

Social interaction does require a social or cultural setting that enables the communication of meaning through signs in the form of sounds, gestures, or written symbols. The knowledge to generate and understand signs consists of a system of signification and a system of sign production (Eco, 1976). Semiotics states that in order to understand the meaning of a sign, the sign has to depend on a code or convention for communication (Jakobson, 1971). Hence, when actors interact, codes provide a medium by which mutual understanding and transaction of meaning between these actors is possible. Chandler (2002) states that:

> [T]he conventions of codes represent a social dimension in semiotics: a code is a set of practices familiar to users of the medium operating within a broad cultural framework [and...s]ociety itself depends on the existence of such signifying systems. (p. 148)

An actor that participates and understands the meaning of signs based on such a convention identifies itself as a member of a society.

The theory of codes or signification system distinguishes different types of codes, such as social codes (e.g. speech or gestures), textual codes (e.g. scientific

[10]We also adopt semiotics as the basis of humans as information-processing entities or cognitive and symbol manipulating actors (see chapter 4).

codes, music) and interpretative codes, e.g. perceptual codes. Codes as systems of signification are rules or sign-functions that establish the correlation of an element of the expression system with the element of the content system. Such a signification system is "an *autonomous semiotic construct* that has an abstract mode of existence independent of any possible communicative act it makes possible" (Eco, 1976, p. 9). Whereas Eco refers to a signification system as an autonomous semiotic construct, we would rather refer to it as a *social construct*—a social signification or coding system. The social construct is based on organisational semiotics, social constructivism and cognitive science; we will discuss the social construct in the next section

In this section, we want to address the transmission of messages between actors and the aspect of sign production[11] that concerns communicative or speech acts. The transmission of messages between addresser and addressee is often portrayed with models of communication processes that describe "the passage of a signal (not necessarily a sign) from a source (through a transmitter, along a channel) to a destination" (Eco, 1976, p. 8). A well-known and often used model is the model of Shannon and Weaver (1949), but although its simplicity is its strength, the model is restricted because it neglects the importance of codes and social context. The model of Jakobson (1960), based on the work of Bühler (1934), addresses the importance of codes and social context and moreover, he states that any act of communication should contain six constitutive factors in order to communicate efficiently and effectively, i.e. (as cited by Chandler, 2002, p. 177):

> The *addresser* sends a message to the *addressee*. To be operative the [*message*] requires a *context* referred to ('referent' in another, somewhat ambivalent, nomenclature), seizable by the addressee, and either verbal or capable of being verbalized, a *code* fully, or at least partially, common to the addresser and addressee (or in other words, to the encoder and decoder of the message); and finally, a *contact*, a physical channel and psychological connection between the addresser and the addressee, enabling both of them to stay in communication. (Jakobson, 1960, p. 353)

Jakobson argues that these six factors are connected to functions of language, see table 3.1.

Besides that, he argues that "these *primitives* (as given by each of the factors and functions) must be (1) present in every speech act and (2) viewed as constantly renegotiating internally their level of importance in any and all speech acts" (Andrews, 2003, p. 18), i.e. although there is a distinction in six functions, verbal messages that fulfil only one of these functions are rare and hardly found (Jakobson, 1960). Sebeok (1991) complements the model of Jakobson, by stating that the *context* is the factor within which the entire communication act is embedded, i.e. the producer and receiver of signs who are involved in the communication act, have their own (built-up) internal represented context—in

[11] For the interested reader, more aspects of sign production, such as a typology of modes of sign production and so on, are elaborated by Eco (1976)

Type	*Oriented towards*	*Function*	*Example*
Referential	Context	Imparting information	It's raining
Expressive	Addresser	Expressing feelings or attitudes	It's bloody pissing down again!
Conative	Addressee	Influencing behaviour	Wait here till it stops raining
Phatic	Contact	Establishing or maintaining social relationships	Nasty weather again, isn't it?
Metalingual	Code	Referring to the nature of the interaction (e.g. genre)	This is the weather forecast
Poetic	Message	Foregrounding textual features	It droppeth as the gentle rain from heaven

Table 3.1: Jakobson's six functions of languages (adopted from Chandler, 2002).

contrast to the external context (including the context of others)—in which they create meaning for themselves. Sebeok relates this to the functional circles in the Umwelt of Von Uexküll, and states that the biological use of Umwelt is not applicable to the explanation of human signs. Besides the Umwelt—the external context—there is an internal context:

> Sebeok was the first effectively to point out... [the difference]... between language, which is a matter of an Innenwelt or modeling system that is not wholly tied to biological constitution, and communication, which is a universal phenomenon that in and of itself has nothing whatever to do with language. (Deely, 2001, p. 132)

Eco (1976) also addresses the importance of the internal and external world by creating a distinction between *codes or signification (system)* and *sign production* or communication.

> There is a signification system (and therefore a code) when there is the socially conventionalized possibility of generating sign-functions, whether the functives of such functions are discrete units called signs or vast portions of discourse, provided that the correlation has been previously posited by a social convention. There is on the contrary a communication process when the possibilities provided by a signification system are exploited in order to physically produce expressions for many practical purposes. (Eco, 1976, p. 4)

This internal and external world is displayed in figure 3.3, a communication model that is a combination of the communication model of Sebeok and the framework of Goldkuhl. The combined model focuses on the communication process, but shows that we do not want to neglect the existence of an internal world of context and its signification system, i.e. it complements the model of Goldkuhl that focuses on the communication process.

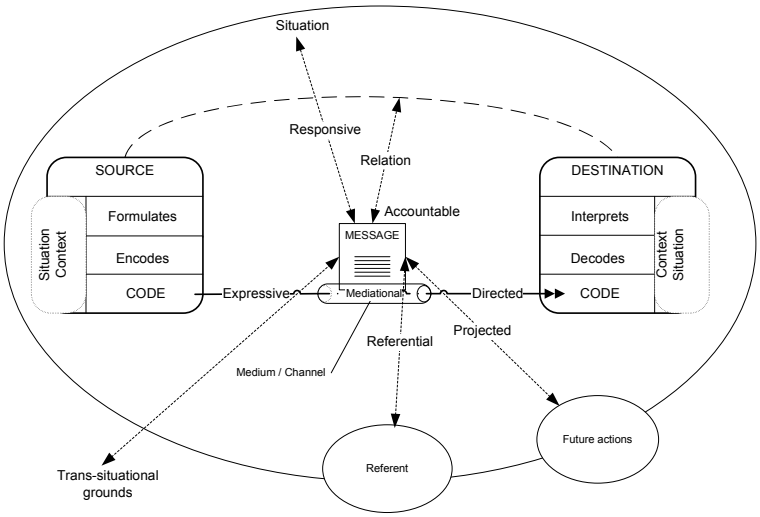


Figure 3.3: Communication model (adapted from Sebeok, 1991; Goldkuhl, 2005).

Goldkuhl's model describes a communication situation (external world) between two actors, the addresser (source) and the addressee (destination). The relation between the message and the addresser is called *expressive* and is *directed* to the addressee. The message functions as a mediator thereby creating a relation between both actors and is referential in that sense that the message *stands for* something. Messages are related to other messages and depend on a social situation, the time and place, i.e. such a relation to prior messages and other circumstantial factors is called situationally responsive. Messages often are directed to future actions or excite a response in someone else, e.g. when a person says to someone else: "It is cold in this room!", then the response can be the closure of a window. This function is called projected and can often explicitly be put as reference in a message, e.g. "It is cold here; could you close the window?". Goldkuhl also refers to trans-situational social grounds, which can be of a general character such as social institutions and norms. The model of Goldkuhl mainly focuses on the communicative acts and not on the underlying signification system or theory of codes. The message is the locus of attention and not the actor and the conventions or social constructs to which an actor itself socially commits, i.e. the model should consider that "*every act of communication to or between human beings*—or any other intelligent biological or mechanical apparatus—*presupposes a signification system as its necessary condition*" (Eco, 1976, p. 9). Hence, the model of Goldkuhl lacks a signification system but it highlights the importance of communication functions and is concerned with what

the message should contain[12], see table 3.2.

Communication function	Explanation	Corresponding question
(1) Trans-situationally compliant	The message is in accordance with general institutions and norms and specific trans-situational grounds, which may be brought into the situation as a social background for the communication.	Why? What reasons there?
(2) Situationally responsive	The message may be a response to prior messages in the communication situation and other situational features.	Why? What reasons here?
(3) Expressive	The message is an expression of the addresser's subjectivity (intentions, desires, skills, emotions, values, understandings, commitments etc).	By whom?
(4) Referential	The message says something about something in the world; objects and processes talked about.	About what?
(5) Accountable	The message needs to be comprehensible, which means that it must to some degree be self-contained and include sufficient self-references and arguments to explain its role in the communication process.	What is meant?
(6) Directed	The message is directed towards one or more addressees. There may also be secondary recipients (an audience).	To whom?
(7) Relational	The message establishes certain action relations (expectations, commitments, states) between addresser and addressee, and sometimes socially broader.	What is done?
(8) Projected	The message may be an initiative for further actions.	For what?
(9) Mediational	The message is expressed in some medium (channel, carrier) and thus utilising the particular features of this medium.	How? By what means?

Table 3.2: Explanations of communicative functions (Goldkuhl, 2005).

The analysis of messages and its communicative functions can support MAS developers in designing a message-based communication model for actors in a Multi-Agent System that interact with the help of these messages. In the field of

[12]The framework of Goldkuhl can be used to analyse (types of) messages. To be concrete, consider a police-officer who writes a ticket to a commuter who violated a traffic-rule. The analysis of the message is that: (1) the message has its trans-situational background in the system of law and norms of conduct in traffic, (2) the message is relating to the current situation of the driver who ignored the red light (3) the police-officer expresses the intention to influence the behaviour and future actions of the driver by ordering him not to ignore the red-light again, (4) the message refers to the law system, i.e. the obligation to follow traffic rules as a driver, (5) the message should be comprehensible, i.e. the ticket its meaning should be clear to the addresser in such a way that the addresser is responsible for its current and future actions, (6) the ticket should contain information (name, address, etc.) that it is directed to that specific driver, (7) the ticket addresses the relationship between the police-officer and the driver; it expresses the roles of the police(-officer) and the driver, (8) the message directs to change in behaviour and payment of the fine, and (9) the carrier is paper, and of course the (non)verbal communication between the police-officer and the driver.

MAS, many agent-application developers mainly analyse or model the expressive function that, according to Goldkuhl (2005), corresponds to the speech act theory (cf. Austin, 1962; Searle, 1969). The speech act theory is concerned with communication as action, i.e. speech actions are performed by agents just like other actions, in the furtherance of their intentions (Wooldridge, 2002). In speech act theory there is a differentiation in functions of speech: the *locutionary*, the *illocutionary* and the *perlocutionary* (Austin, 1962). They represent, respectively, the making of an utterance (what is talked about), what is done by the locutor in saying something, and what the effects are on the addressee. Searle (1979) divided the illocutionary function into a classification of five main types of speech acts that are commonly used in the field of Information Systems and MAS:

1. *Representative act* which commits the speaker to the truth of the expressed proposition (inform, asserting, concluding).
2. *Directive act*, which is an attempt by the speaker to get the addressee to do something (requesting, questioning).
3. *Commissive act*, which commits the speaker to some future course of action (promising, threatening, offering).
4. *Expressive act*, which expresses a psychological state (thanking, apologising, welcoming, congratulating).
5. *Declarative act*, which effects immediate changes in the institutional state of affairs (declaring war, christening, marrying, firing from employment).

Speech act theories, like the ones of Austin and Searle have had an impact on the development of actor communication languages in Artificial Intelligence and Multi-Agent Systems.

Actor Communication Language[13]

In order for actors to be social, exchange knowledge and have meaningful, constructive and intelligent interaction, the actors should not only have knowledge of a common language, but also a common understanding of the terms used in a given context. Thus, as mentioned before, the actors require a social convention or common language and a signification system (common understanding).

For communication and shared understanding to be effective, the actor requires a couple of components that are necessary for exchanging knowledge. Figure 3.4 shows the possible components of an actor: the representation components, the communication components and components that are not directly related to shared understanding (Finin, Fritzson, McKay, & McEntire, 1994). According to Finin et al. (1994), the communication can be divided into knowledge of an (1) interaction protocol, a (2) communication language and a (3) transport protocol.

[13]We could have discussed this section in chapter 2, but addressing it in this section allows for a more convenient comparison with semiotics

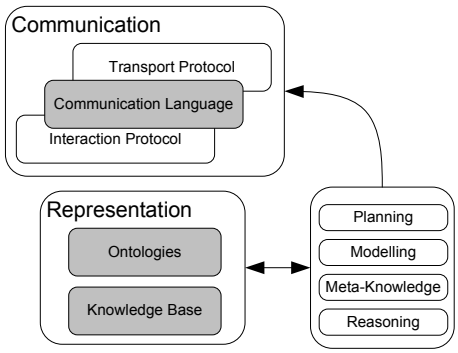


Figure 3.4: Components of the actor (adopted from Finin et al., 1994, p. 3).

The interaction protocol refers to high level strategies that governs interaction with other actors and are context sensitive, e.g. a negotiation scheme or a social construct that guides the communication: "If that actor asks me something I first require permission of my boss". The interaction protocol is a social construct; a construct on which we will elaborate more in section 3.4.2. The communication language (e.g. KQML, explained later) however, does not involve the social situation, but is more related to attitudes regarding the content of the exchange communicated, i.e. if it is a request, assertion or any other form of query (Finin et al., 1994). The transport protocol is the physical network protocol (TCP, HTTP, FTP), which we leave out of the discussion in this dissertation. Similar to this distinction is the semiotic ladder of Stamper (1973) that states that communication is built up from the physical world towards the social world, see table 3.3.

<table>
<tr><th colspan="2">Stamper's semiotic ladder</th><th>Finin et al.</th></tr>
<tr><td>Social world:
norms, contracts, law, culture...</td><td rowspan="2">Human Information Functions</td><td rowspan="2">Interaction Protocol</td></tr>
<tr><td>Pragmatics:
Intentions, communication, conversations...</td></tr>
<tr><td>Semantics:
meanings, signification, validity...</td><td rowspan="2">System Platform</td><td rowspan="2">Communication Language</td></tr>
<tr><td>Syntactics:
formal structure, logic, data...</td></tr>
<tr><td>Empirics:
variety, channel capacity, codes...</td><td rowspan="2"></td><td rowspan="2">Transport Protocol</td></tr>
<tr><td colspan="2">Physical world:
signal, traces, hardware...</td></tr>
</table>

Table 3.3: Similarity between two approaches of communication (adapted from Stamper, 1973).

The [physical world] and empirics are concerned with the physical aspects of signs and their statistical properties when different

> media are involved. Together with syntactics, they constitute the infrastructure to support the other three layers: semantics, pragmatics and social world. The three lower layers will answer questions referring to how signs are organised and expressed, as well as the physical properties they have, while the upper layers [semiotic layers] are concerned with the use of signs, how they function in communicating meanings and intentions, and the social consequences of using signs. (Baranauskas, Liu, & Chong, 2003, p. 69)

As Baranauskas et al. (2003) acknowledge, there is still a lot of discussion about the distinctions made between social world vs. pragmatics and pragmatics vs. semantics. In the discussion of this dissertation, we therefore will stick to the distinction of Finin et al. (1994) that will suffice in the development and use of Agent Communication Languages as implemented in our MAS in chapter 5.

In order to communicate, actors have to share knowledge and the concept of communication should be used in the sense of *sharing*, i.e. a process of signification shared by a community or according to Cherry (1980, p. 30) "communication is an act of sharing". For actors, to share information they are assumed to have representations[14], such as a knowledge base. A set of rules regarding syntax and semantics that can give structure to the elements in the knowledge base is often referred to as an ontology. According to Wooldridge (2002, p. 180):

> An ontology is a formal definition of a body of knowledge. The most typical type of ontology used in building [actors] involves a structural component. Essentially a taxonomy of class and subclass relations coupled with definitions of the relationships between these things.(Hendler, 2001)

The knowledge base and ontology can be expressed in a content language, such as KIF (Knowledge Interchange Format) or XML (eXtensible Markup Language), in which the declaration of properties of a domain and the relationships between things in the domain can be expressed. Whereas KIF and XML are concerned with the content, KQML (Knowledge Query Markup Language) is an 'outer' language (the envelope) that encompasses the content language. The KQML language can be thought of as consisting of three layers: the content layer, the communication layer, and the message layer (Finin et al., 1994).

> The content layer bears the actual content of the message, in the [actors'] own representation language... The communication level encodes a set of message features which describe the lower level communication parameters, such as the identity of the sender and recipient, and a unique identifier associated with the communication. It is the message layer that is used to encode a message that one application would like to transmit to another. A primary function of the message layer is to identify the protocol to be used to deliver the message and to supply a speech act or performative [,e.g.

[14] And of course cognition, e.g. cognitive mechanisms that can handle representations.

> one of the five types defined by Searle,] the sender attaches to the content... In addition, since the content may be opaque to an actor, this layer also includes optional features which describe the content language, the ontology it assumes, and some type of description of the content (Finin et al., 1994, pp. 10–11).

We take an example of a KQML dialogue, see table 3.4, to illustrate its working in interaction between actors.

KQML Dialogue

```
Message X
(evaluate
                :sender A :receiver B
                :language KIF :ontology motors
                :reply-with q1 :content (val (torque m1))
)
Message Y
(reply
                :sender B :receiver A
                :language KIF :ontology motors
                :in-reply-to q1 :content (= (torque m1) (scalar 12 kgf))
)
```

Table 3.4: KQML dialogue (Wooldridge, 2002, p. 173).

Actor A sends a message (Message X[15]) to actor B with a query (q1) that requests the value of the torque on m1. In the message, the KQML performative is *evaluate*, the content is *(val (torque m1))*, the ontology used in the content is *motors* and the language is *KIF*. Actor B sends a response (reply) that the *torque m1* is *12kgf*—a scalar value (Wooldridge, 2002).

KQML is an Agent Communication Language (ACL), based on speech acts theory. FIPA[16] (FIPA, 2002) is a common standard Agent Communication Language, superficially similar to KQML, and an example of an agreement—a signification system—between agent-application developers about how agents in a MAS should communicate with each other, i.e. they should have the same textual coding system (e.g. FIPA) in order for agent-applications to 'understand' each other.

However, in MAS, social interaction is often only implemented based on the communication process in the form of speech acts (such as KQML) and tries to describe the social level of actors with help of this communication process. In our opinion, semiotics gives social interaction a better theoretical ground in the form of a signification system that resides mainly in the minds of actors. In the next section, we want to discuss the social actor, social constructs and the semiotic level that describes the use of conventions, norms and signs in communication and interaction with other actors.

[15]The three levels of KQML can be discerned in Message X; the *content* keyword is the content level, the values of the *sender*, *receiver* and the *reply-with* are part of the communication level and *reply*, *evaluate*, *language* and *ontology* form the message layer.

[16]The Foundation for Intelligent Physical Agents

3.4 Semiotic level, social constructs and the social actor

Artificial Intelligence (AI) has made progress in trying to understand its connection with the physical world and lately more specifically with the social and cultural world. AI started with the classical cognitive approach that describes an actor based on symbol systems and states that this is a sufficient and necessary condition for intelligence (Newell & Simon, 1976; Newell, 1980). However, the approach focuses on the internal organisation of the actor (the cognitive system) and not (so much) on the influence of the outside world on the functioning of the cognitive system. Later on, in the nineties, *embodied cognition* or *situated cognition* gained a lot of popularity and defines that the actor's bodily experiences are acquired through its interaction with its environment, which is also referred to as *physical situatedness*, or *'New AI'* (Suchman, 1987; Brooks, 1991b; Clark, 1999). 'New AI' argues that symbol representations are not always necessary to implement intelligent behaviour.

Both approaches have delivered some wonderful applications we could never have dreamt of, but have a disadvantage: the problem of paying no attention to the social and cultural setting in which these applications function. The disadvantages of the classical approach is that the symbol is loosely connected to the outside world and therefore the symbol in the symbol system doesn't acquire its meaning from reality, referred to as the *symbol grounding problem* (Harnad, 1990). The problem with *New AI* is that it is able to implement many (lower) cognitive functions, e.g. obstacle avoidance, attraction by light, but that the higher cognitive functions such as language processing and self-reflection are not possible without any representation (Vogt, 2002).

3.4.1 The situated actor and the semiotic level

Human beings can be considered to have some degree of *physical situatedness*, but it has been argued that humans are also *socially* (and *culturally*) *situated* (Lindblom & Ziemke, 2003). Dautenhahn et al. (2002) suggest to broaden the concept of (physical) situatedness; it not only includes the physical environment but the social or cultural environment as well. Lindblom and Ziemke (2003) state that the Russian scholar Lev Vygotsky (1962, 1978) (together with Mead, one of the first social constructivists) already pointed out the importance of social interactions for the development of individual intelligence during the 1920-1930s. Vygotsky argued that individual intelligence emerges as a result of the biological factors in combination with interaction with a social environment through a developmental process.

Hence, there is a need to connect the social, cultural and physical world with the 'Innenwelt' of the actor. We suggest a separate level of description, the semiotic level that comprises both, the Umwelt and the Innenwelt of the actor and creates a social actor that is both embodied and situated, i.e. the semiotic level

is a level of signs or constructs that connects the cognitive or individual level[17] with the social level.

As stated in chapter 1, the semiotic level describes the use of language and signs in communication and interaction in order to agree on social constructs (e.g. common plans, or contracts) (Gazendam, 2004; Helmhout et al., 2004; Helmhout, Gazendam, & Jorna, 2005a). We adopt theories of semiotics and more specifically organisational semiotics in order to bridge the gap between the social level and the individual level. The social constructs at the semiotic level—explained in the next section—can be represented as signs for supporting social interaction between actors, and also as representations at the individual or cognitive level in the form of symbols or chunks in a physical symbol system (Newell, 1980).

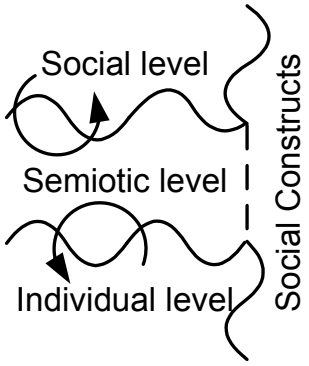


Figure 3.5: Social constructs present at three levels.

Figure 3.5 shows that the representation of social constructs is present at the different levels of description. At the social level, social constructs are stored in organisations or institutions as (social) artefacts, at the semiotic level, social constructs are represented as signs and at the individual level, social constructs are represented in the form of chunks that are stored in a physical symbol system. As mentioned before, semiotics is present at all three levels. Therefore, we have drawn the curved boundaries between the levels. The semiotic level its function is to mediate between those levels (with help of its signification system and sign production) and allows the sharing of social constructs between the social and individual level.

The discussion about the cognitive agent and the physical symbol system will be elaborated in the next chapter. In the remaining part of this chapter we want to focus on the semiotic level and social constructs, and why they are required for an actor to behave socially.

[17] Actually there are five levels; the individual level can be divided into three levels: the intentional, functional and physical level (cf. Dennett, 1978, 1987), but we want to keep them out of the discussion in this section.

3.4.2 Social constructs

As mentioned before, social constructivism states that a social construct can be seen as an invention or artefact constructed by interaction between members of a social group or interaction between groups. The aim of social constructs, as proposed in this chapter, is to bridge the gap between the social level and the individual level of the actor and finds its origin in organisational semiotics—the Stamper school—that suggests the combination of *affordances* and *signs* to bridge this gap (Stamper, 1973, 2001).

> Affordances stress the interaction between a human agent and its environment based on behaviour patterns that have evolved over time in a community. Signs stress the social construction of knowledge expressed in sign structures... Stamper sees affordances as repertoires of behaviours and distinguishes physical affordances and social affordances. (Gazendam & Liu, 2005, pp. 7–8)

A (physical) *affordance* is a set of properties of the environment that makes possible or inhibits activity (Gibson, 1979). After many encounters with the environment, this can result in a *habit of action*, which is a commitment to act with a connected action program that governs the actual acting (Peirce, 1931; Gazendam, 2001, p. 40). From a semiotic point of view, one could say that a physical affordance becomes a *social affordance*[18] as well, the moment the physical affordance is shared between actors in a community (e.g. a traffic light). The experience of the object (shared with others) is built up in the mind of the actor; the actor is socially situated through interaction and perception (e.g. of the traffic light), which is a process of social construction of signs in the actor's mind. The resulting signs are organised as units of knowledge consisting of a *semi-indexical*[19] representation of an affordance and its associated habit of action.

Social constructs are social affordances (Liu, 2000; Stamper, 1973, 2001) and can be seen as representations of cooperation and coordination, based on intertwined habits and mutual commitments that are often expressed in sign structures such as agreements, contracts and plans. A social construct (Gazendam, 2003; Liu, 2000) is a relatively persistent socially shared unit of knowledge, reinforced in its existence, by its frequent use. In organisations, social constructs take the form of, for instance shared stories, shared institutions (behaviour rule systems), shared designs, shared plans, and shared artefacts. These social constructs support habits of action aimed at cooperation and coordinated behaviour.

More or less following Dignum (1996)'s stratification of social norms, social constructs can as well be differentiated in types or levels of social constructs:

[18] The social affordance has a strong social basis and could be defined as 'a pattern of behaviour that is shared in a social community', to be distinguished from Gibson's affordance concept that is connected with perception and is defined as 'characteristics of the world' that afford a certain pattern of behaviour (Gazendam & Liu, 2005).

[19] "A semi-indexical... [representation]... results from a process where raw data resulting from the interaction between actor and environment are unconsciously filtered, transformed, and matched to patterns (Marr, 1982; Jorna, 1990, p. 57)." (Gazendam, 2003, p. 2)

the first type is the multilateral institutional or behavioural system in which actors commit to social constructs of a community, organisation, country or society. These constructs are formalised and mostly written down in documents and are part of a large population; they are 'institutionalised' in order to coordinate behaviour between people. Many of these constructs formed at this level are constructs that prescribe patterns of behaviour in other social constructs (its descendants) and are general indications; they state in general what is allowed and what not. An example of a multilateral social construct is the freedom of speech: the question if scolding and thereby harassing people can still be considered as falling under the law of freedom of speech is debatable and whether strict written rules can cover this problem. Therefore social norms play such an important part in society by establishing tacit codes or patterns of behaviour, thereby filling the gap that cannot be covered by rules.

The second type is the plan or model to which individual actors or groups are committed to. They regulate appropriate or coordinated behaviour for smaller groups. For instance, planning and teamwork models (Decker, 1995; Jennings, 1992) are social constructs that consist of joint intentions in which actors reach agreements about how tasks should be allocated and planned, according the schedule they have committed to.

The third type is the bilateral construct established between two actors, e.g. a marriage contract or a producer-customer relationship in which there is an agreement about the rules of behaviour during the life-span of the relationship.

These three types of social constructs can be captured into an abstract or generic theoretical framework of social constructs that has properties applicable to all these different types of social constructs.

In this dissertation, we apply the social construct[20] as a social construction established by interaction of actors at the social level. In the discussion about the semiotic level, we have stated that social constructs are not only part of an organisation, but also (according to methodological individualism) are represented as (shared) knowledge in the mind of actors. In order to use the social construct (as a knowledge chunk) in an actor model, there is a need to distinguish and formalise properties of the social construct.

We have defined the following theoretical framework of (a not limited set of) properties of a social construct:

Attached norms or rules: social constructs can contain (a collection of) norms or rules that guide action and prescribe appropriate behaviour in a certain context. *Norms*[21] can be defined as rules of conduct that constrain (self-interested) behaviour and that are adopted and enforced in a (in) formal setting (Mahoney & Sanchirico, 2001). Our daily encounters with social norms (and law) are evident, for instance, when we are driving with our car at the right side of the street, or being polite for letting the elderly sit in the bus, etc.

[20] "According to the Stamper school of organisational semiotics (Stamper, 1973) there are main social constructs called *social affordances*, to which norms are attached". (Gazendam, 2006, p. 158)

[21] There is a large bulk of definitions and literature on norms, a complex field in sociology, law and economics which is not treated here in detail. (see Gibbs, 1965; Etzioni, 2000; Morris, 1956; Sunstein, 1996; McAdams, 1997)

Written/unwritten (coded/sensory): a social construct can be formed and communicated by writing the attached rules and norms down on paper, or they are internalised in actors and with the help of interaction transferred (language or gestures) to others (March et al., 2000).

If we refer again to the traffic law, then this is an example of a written social construct that clarifies (1) an evaluation of what actions or behaviours are (not) permitted and (2) the type of reactions to the (incorrect) behaviour, such as sanctioning or rewards.

In the case of the bus and an old woman[22], the social construct and its norms are not registered with an authoritative organ, instead, behaviour is controlled and reinforced by collective expectation and evaluation of others and assumptions about possible reactions that eventually can result in shame and guilt.

Life span: every social construct has a starting time, an evolution, a monitoring and controlling period and a finishing time (Liu, 2000, p. 67). The life span of every social construct, be it a norm, an emotion or an organisation varies and depends on other properties connected to the social construct, e.g.

- One of the referred *objects* or *facts* changes in such a way that it falls under another categorisation or role, e.g. when a person dies, it falls under the category of dead people and cannot fulfil its expectations anymore, or an organisation has gone bankrupt, in which case all connected social constructs (and norms) are not linked to the organisation anymore, or when a child grows up to become an adult, it becomes financially responsible for its own behaviour, and so on.
- Lack of *reinforcement* of the social construct will let the social construct disappear, e.g. if a social construct is not communicated to others and violations not corrected, then the construct will never become a habit and will never become part of society.
- Changes in *enforcement costs*: a social construct is often kept in place because people enforce others to obey the norms attached to the social construct. McAdams (1997) gives an example of an anti-abortion movement that informally monitors clinics and sanctions those women that enter the clinic. "[]. . . a new drug. . . allows women to obtain abortions with visits to their regular doctor rather than a clinic, which in turn, raises substantially the cost of detecting an abortion" (pp. 394–395). Detection of abortion and the enforcement of anti-abortion becomes too expensive and gradually more women will go to a doctor to get an abortion and the anti-abortion movement probably will loose ground.

Roles and identification: the actor is given a role or identification, e.g. employer, employee, to make clear the authority, control and rules applied

[22]There is sometimes even a distinction in norms and gender, i.e. letting an old woman sit and not the old man, because the old man is assumed to be stronger than the woman.

to that role (Liu, 2000). Roles are often associated with norms, i.e. "... roles are...[]...a product of social norms" (Sunstein, 1996, p. 928). These roles can be common roles like 'the citizen'—how does a good citizen behave—or to a specific function like a general of an army. Panzarasa, Jennings, and Norman (2001) state that adopting a role imposes responsibilities, influences and constraints upon the role-player: a) a set of expectations about the other role-players' mental attitudes and behaviour and b) a set of expectations of how the other role-players will respond to the role-player its own mental attitudes and behaviour. The difference between roles is often regulated by the following property: authority, responsibility and control.

Authority, responsibility and control: according to Fayol (1918), authority can be seen as 'the right to give orders' and the expectation that they are followed. Control and power can assure that actors behave responsible; they can be part of a directly involved authoritarian party or an independent third party. Assigning authority to someone creates a responsibility for that person to give orders and control whether other actors take their responsibility in following the 'rules of the game'. Responsibility also means that someone with authority has a duty to report about his/her behaviour to the community or authority that has given him/her the authority; Fayol remarks that authority is sought after, but that responsibility is shunned.

Stamper (2001) argues that authority is necessary to start or finish a social construct. For instance, a police-officer who has the right to regulate traffic, has the authority to finish (temporarily) the social construct of obeying to the signals of the traffic-lights and instead starts of a social construct in which the signals of the police-officer replace the previous social construct of traffic lights[23].

Inheritance or prerequisite of other social constructs: a social construct can be part of a complex network of connections with other constructs (that are often the result of earlier agreements). For example, when preparing a sales contract, sales men refer to their conditions that are registered at the institution of commerce and the registered conditions inherit conditions from public law (Liu, 2000, p. 68).

Scenario: according to the language action school (Barjis, Dietz, & Liu, 2001), there can be a more or less standardised process (scenario) for establishing a social construct between actors. Scenarios are often put on paper in which a specific order of social constructs over time is written down. In communities, scenarios are expressed in rituals and transferred from generation to generation.

The scenario is similar to the concept of script of Schank and Abelson (1977), which they define as: "A script is a structure that describes appropriate sequences of events in a particular context. A script is made up of slots and requirements about what can fill those slots. The structure is

[23]Such a scenario is actually authorised by another law (authorised by the justice department) that says that signals of a police-officer overrule all other signs of traffic law.

an interconnected whole, and what is in one slot affects what can be in another [p. 41]" (as cited by Jorna, 1989). A well known example of a script or scenario is the 'restaurant script' of Schank and Abelson that describes the typical scenario of ordering food in a restaurant.

Another example of a scenario (or script) is a marriage in the Netherlands. It has to be prepared by well defined pre-conditions such as pre-marriage in which case the couple has to be registered for at least two weeks and a maximum of a year. The actual marriage is often a combined marriage, one of a religious character and one of a governmental character. Besides the registration issues, the scenario takes preparations from both sides in the form of norms concerning clothes, the dowry, the witnesses, and so on.

Context: context as property is a debatable issue, however there are two interpretations of context. Context can be situated *outside* the actor, and—possible at the same time—situated *inside* the actor, i.e. context is represented as symbols in the 'real' world stored externally from the mind and also as symbols stored in the mind of the actor.

The external context contains certain (impelling) elements—so-called affordances (Gibson, 1979)—perceived by the actor to which it is sensitive or is triggered by. In contrast to Gibson, and similar to Stamper (Stamper, 1973; Stamper, Liu, Hafkamp, & Ades, 2000), Vera and Simon (1993, p. 41) state that affordances "are carefully and simply encoded internal representations of complex configurations of external objects, the encodings capturing the functional significance of the objects".

We assume that there has to be sufficient coherence between an internally represented social construct and an element in the external environment in order to activate or trigger the social construct. According to Gazendam, Jorna, and Helmhout (2006):

> The recognition of a situation in which a [social construct] must become active must not only depend on the recognition of physical affordances like other actors, objects and situations, but also of an ongoing monitoring in the actor's mind of the state of the social context in terms of invisible entities like positions of rights and obligations of himself and of other actors, agreements and appointments that must be held, and so on. (p. 2)

To demonstrate what impact context can have, compare the police-officer signalling directions for traffic and the same police-officer taking part in a play in a theatre signalling the same directions for dancers. The only difference in both cases is that the context is different. In the case of the theatre, the dancers, spectators, spotlight, and so on, create a different atmosphere in which the police-officer cannot function as a 'real' police-officer anymore. The environment (dancers, spectators etc.) and its associated expectations do not create the same affordances for the police-officer.

Social constructs with all their properties are internalised and established by a process of socialisation and communication, and during their life-span are

monitored by enacting actors, through observation, mentoring, practise, and training (March et al., 2000). Secondly, they create standards of appropriate behaviour and stabilisation, i.e. they create shared expectations of behaviour and when exercised, they are evaluated by others as well. Thirdly, when they are widely known and accepted as legitimate, they are often self-enforcing, and its associated psychological cost of rejection will rise, e.g. the individual or group involved can feel psychological discomfort whether or not others detect the rejection (McAdams, 1997). And fourthly, because social constructs are connected to roles and authority, they can create formal and informal social structures.

Such a social structure exists out of a network or collection of social constructs. When a social structure is legitimate, legally acknowledged and officially registered, it is often referred to as an organisation or institution. The form and life-span of a social construct depends on change of context, entering and exiting actors, change of roles and authority and many more possible modifications of properties. Some constructs have shown a long experience of successes and are written down on paper, thereby encoding parts of history; they accumulate experiences generation after generation.

In chapter 6, this theoretical framework will be applied for analysing two demonstrative experiments. Both experiments are concerned with traffic laws and coordination between two actors. The first experiment demonstrates the emergence of social constructs as a result of adaptation and learning. The second experiment is concerned with actors that have prior knowledge about social constructs and the transfer of social constructs between actors. In the second experiment, the actors are more organised in the sense that one of the actors is assigned the role of police-officer and the other the role of citizen.

3.4.3 Social constructs and organisation

An organisation is a collection of actors that share a collection of social constructs and every actor is taking part in one or more situations that are regulated by these shared social constructs. However, each actor has individual properties, and experiences different interactions, perceptions and cognitions during its life inside and outside the organisation. Hence, the actor possesses personal (also biological) characteristics, e.g. attitudes, norms and values. Personal characteristics and experiences (the so-called personal social construct) have influence on the creation of different representations of social constructs in the heads of the actors, thereby creating different representations of the organisation as a whole in the mind of every actor, i.e. the participation (now and in the past) of the actor in several communities accounts for an unique personal collection of social constructs. Based on individual experiences and interaction with other actors, every actor can build up a *unique* history of social constructs that are successful or unsuccessful and can act on its environment based on that experience.

Figure 3.6 shows that two actors, X and Y in a context or organisation, share knowledge in the form of internalised social constructs (representations in the mind) and externalised social constructs (e.g. physical artefacts and documents such as contracts, rules or laws). The actor has a personal (social) construct that is partly in coherence with the social constructs of the environment, i.e. he com-

mits himself to the social constructs that overlap with what the actor thinks he has in common with the social constructs in its environment. In the case of social constructs that do not overlap with his personal (social) constructs, there can be for example a mechanism that excites some (emotional) resistance in following those, i.e. the actor follows the social construct, but his personal preferences are in favour of not doing so; he is forced by the social pressure of others to follow them. Secondly, there is dynamics of social constructs triggered by language action, change of signs, context or interpretation differences. The interaction of signs is a process of semiosis in which actors express their meaning with help of a sign production and an underlying signification system.

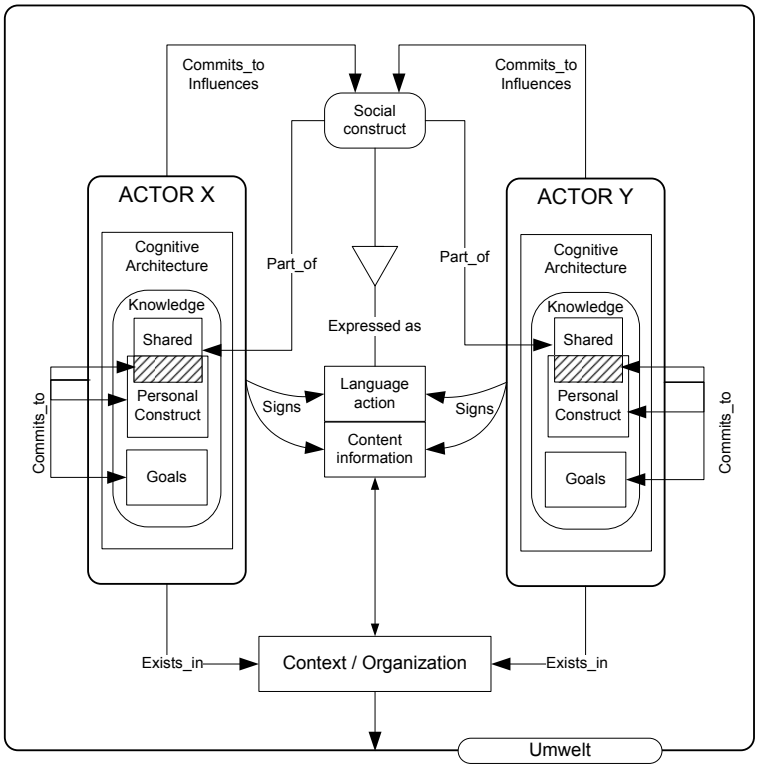


Figure 3.6: Social constructs, actors and context (adapted from Helmhout et al., 2004).

We argue that social constructs are formed by interaction of actors and the process is based on individual learning and personal (social) construction by each actor involved, i.e. there is an establishment of the social construct in agreement with others and at the same time this social construct is internalised in the mind of the actor. Argyris and Schön (1978, pp. 16–17) similarly argue that individuals jointly create public maps, but at the same time have private images of that map:

> An organization is like an organism each of whose cells contains

> a particular, partial, changing image of itself in relation to the whole. And like such an organism, the organization's practice stems from those very images. Organization is [a social construct] of individual ways of representing organization. Individual members are continually engaged in attempting to know the organization, and to know themselves in the context of the organization. At the same time, their continuing efforts to know and to test their knowledge represent the object of their inquiry. Organizing is reflexive [or reflective] inquiry...[Members] require external references. There must be public representations [, i.e. social constructs] of organizational theory-in-use to which individuals can refer.[...]These are the shared descriptions of the organization which individuals jointly construct and use to guide their own inquiry...Organizational theory-in-use, continually constructed through individual inquiry, is encoded in private images[—personal constructs—] and in public maps[—social constructs]...

The existence of public maps or shared social constructs are only possible when actors recognise and acknowledge them. Some social constructs are informally established and present in a tacit form while other social constructs are often arranged by authoritative rituals and processes of internalisation and after mutual acceptance, social constructs are maintained and reinforced by authoritative signs (e.g. a law or an authoritative leader gesturing the recorded signs).

In the first case, we rather speak of an *informal* organisation in which social constructs are based on habits of (inter) action, whereas in the latter we speak of a *formal* organisation that maintains a public system of rules, also known as an *institution* that has been assigned social power and authority by civilians that are part of a society. An institution is a *behavioural rule system* (Gazendam, 2003) containing rules, laws and constructs that states how to act in certain circumstances or according to Rawls (1971):

> Now by institution I shall understand a public system of rules which defines offices and positions with their rights and duties, powers and immunities, and the like. These rules specify certain forms of action as permissible, others as forbidden; and they provide for certain penalties and defenses, and so on, when violations occur. As examples of institutions, or more generally social practices, we may think of games and rituals, trials and parliaments, markets and systems of property. An institution may be thought of in two ways: first as an abstract object, that is, as a possible form of conduct expressed by a system of rules; and second, as the realization in the thought and conduct of certain persons at a certain time and place of the actions specified by these rules. (p. 47)

In this section, we have described the properties of the social construct and that an organisation can be seen as a collection of social constructs, i.e. we elaborated more or less about the social structure of organisation and society. However, the processes that create changes in social structure, i.e. the process of cre-

ation, evolution, control and ending of social constructs need to be addressed as well.

3.4.4 Social constructs: creation, evolution, control and ending

The processes of creation, evolution, control and ending are social processes; they are only possible when there is mutual agreement of other (involved) actors (e.g. one cannot change the law by itself without allowance of others). The life span—(in)formally from the moment social constructs are created to the moment they are declared invalid or are neglected—depends on the use of them in social interactions with others, i.e. social constructs are shared and agreed upon between actors. Secondly, in case of the individual, the life span of the social construct depends on the actor's own habits of promoting and using the social construct. The actor's physical limitation of remembering them and/or communicating the construct (in)directly to others influences the life span of social constructs. The assumption is that internalised social constructs are part of the memory of the actor, so their existence is subject to the way they evolve (forgetting and reinforcement) in memory and the (triggered) awareness of the actor about them inside its own memory.

Thus, we argue that social constructs are social in the sense that they are part of a society and its history, and that actors have subscribed or agreed upon the establishment of those social constructs. On the other hand, we ascribe the social construct to the individual level as part of memory, i.e. the actor has internalised the social construct; the cognitive system of the actor determines whether a social construct is active, suppressed or no longer active/forgotten.

In this section we want to illustrate how social constructs and its attached norms do start, evolve, are controlled and end. To make clear that almost any object (a social construct) in nature that is discovered by people have norms attached to it, we consider property and ownership rights (cf. Gazendam, 2006). A well known access right to property is the option on access or ownership of property in the future, i.e. the moment something is discovered people issue options for ownership.

We will illustrate the creation, evolution, control and ending of a social construct with help of an example that explains the sale of options on acres on the moon which is just one of many examples in which people declare their claims on properties. The following citation from "lunar-registry" gives an example in which it claims to be the legal organisation for registering property[24].

> ...your property record[, a claim on property of the moon,] is permanently and publicly registered by the International Lunar Lands Registry in its annual publication, which is copyrighted and deposited in the United States Library of Congress and with international patent and trademark offices, including the United Kingdom (UK Patent Office), Japan (Japan Copyright Office), Russia (Rospatent) and with the United Nations (UN Depository Library), in compliance with the Berne Convention

[24]Retrieved 26-03-2006 from http://www.lunarregistry.com/info/deed.shtml

The idea of claiming "shares" on the moon originally dates from the moment the United Nations "... set forth a plan to protect the Moon and all outer space from exploitation by nations that would seek to claim extraterrestrial bodies in their own name..."[25], i.e. the "Treaty On Principles Governing The Activities Of States In The Exploration And Use Of Outer Space, Including The Moon And Other Celestial Bodies". The twentieth July 1999, the "Lunar Republic" wrote a proclamation of property on the moon and started selling registration certificates on the Internet. From that moment the social construct, i.e. a share of the moon, is started.

Now, we will first analyse the creation of the social construct—the "lunar certificate"—and its properties attached that 'legally' identifies it as a sign of ownership of the moon. Before the process of changing ownership between the 'legal' organisation that distributes lunar shares and the individuals that buy the shares, the organisation has to announce its legitimacy with help of significant symbols in society.

First, when the company—lunar registry—starts up its business, it registers itself with well respected institutions, e.g. the United States Library of Congress, the UK Patent Office, and so on, in order to announce that it inherits rules and norms of these institutions and behaves accordingly to these norms. Second, it claims to be part of a large organisation, the Lunar Republic Society that sells lunar articles and publishes a lunar journal. Third, it claims that is the *only legal* institution for selling shares of the moon, it signals this by a sign that states "officially authorised, Lunar Republic Society". And finally the organisation is trying to convince people that the money of the project goes to the "Kennedy II Exploration Project (K2LX)", a project of scientists, students and post-graduates.

The organisation has inherited social constructs and attached norms of other institutions, it attempts to put itself in the context of and associates itself with these institutions. The social construct "lunar-certificate" is a written document that claims to inherit those properties as well, i.e. being a product of a legitimate organisation. The organisation does not give any information regarding the life-span of the certificate, e.g. what happens when I die, probably the lunar-certificate its expiration is based on the assumption of the life-expectancy of the company self. The purchase of the certificate and the transaction are social constructions as well. The transaction and its rules are according the rules of credit-card companies and the additional rules applied by the organisation itself. Again, in case of transaction by Internet, the lunar-organisation attempts to signal its trust by a trading sign; they register itself and inherit properties of the company 'Squaretrade' that in its part identifies itself with the organisation 'etrade.org'; an independent non-profit organisation enabling trust based on privacy for personal information on the Internet.

A relatively simple social construct such as a lunar-certificate needs and creates many properties associated with a social construct. This example shows how complicated a social construct can be. During the life-span of the lunar-certificate the following processes are distinguished:

[25]Retrieved 27-03-2006 from http://www.lunarregistry.com/info/legal.shtml

1. *Creation*[26]
   The creation of the lunar-certificate is a process of advertising its existence by Internet and at the same time referring to legal institutions to create legitimacy for its existence.

2. *Evolution*
   During its existence, the lunar-certificate evolves, e.g. when other organisations such as moonshop.com are competing for existence, 'lunar-registry' has to change its identification in order to show that it is superior to its competitors.

3. *Control*
   Control of the 'official' registration seems to be impossible, however the organisation claims it is possible because it registers the certificates with legal institutions. In the case of the transaction, again there is no guarantee of being trustworthy, no matter what the 'Squaretrade' sign tries to signal, there is only the assumption that there is a control of trustworthiness.

4. *Ending*
   The ending of the life-span of the lunar-certificate is also not evident. The lunar-certificate is a relationship between the registrar and the owner of the certificate. The moment the registrar or the owner ceases to exist or dies, the relationship ends and the lunar-certificate does not 'legally' exist anymore.

The life span itself has three phases, the creation, the evolution and the ending of a social construct. The control function is a separate process that can be applied during the entire life-span, i.e. during creation, evolution and ending of the social construct. The process of *creating* a social construct can occur in three ways (Flynn & Chatman, 2003):

1. Based on adaptive emerging social behaviour; when two actors have a conflicting or mutual goal/interest. For instance people going out of the elevator while others only can go in when it is empty, or a group meeting in which people after a couple of times play their (habitual) role.
   (tacit agreement in a community)

2. The other is communicative action or an authoritative ritual in which with help of communication, e.g. speech acts, a social construct is formed; it starts with the actor that wants to propose, inform, or is requested to inform or propose a social construct to another actor.
   (agreement in a community)

3. The third formation process is external norm formation and a clear specification of roles, in which certain individuals are authorised, e.g. police officers, to prescribe and enforce norms, thereby regulating the individuals they supervise (Von Hayek, 1979).
   (creation on authority)

[26]There are only two valid ways of creating a social construct: (1) agreement upon it in a community; (2) creation by an appropriate authority (e.g. the United Nations).

While the last two are part of an explicit process in which the actors communicate or bargain openly about what they want, the emergent (informal) social construct creation is an implicit process in which assumptions are made of other actors' behaviours.

After forming the social construct and its attached norms, a process of *evolution* takes place. During evolution, the actor and community learns if the norm is still beneficial to him or his community and if other norms are involved as well. During evolution of norms a complete network of norms can be formed. López y López, Luck, and d'Iveron (2005) mention that norms are not isolated from each other, but are often interlocked—in our approach a social construct—and in which one norm is triggering another norm, e.g. the expectation of an invitation triggers the norm of bringing a gift. The network of norms becomes an informal institution as a system of habits, rituals, norms and scenarios (similar to the scripts of Schank and Abelson (1977) as mentioned before).

The *ending* of a social construct can happen during the process of evolving, when an actor wants to re-negotiate, releases and renews the social construct before the agreed time-span is reached; the contents or time-span are changed, e.g. the announcement of the European Commission of a draft directive on driving licences[27] states that the national driving licenses will be replaced in order to improve "the free movement of citizens by ensuring mutual recognition of licenses by the Member States". Secondly, when the agreed time-span of a social construct ends, the involved actors mutually release the social construct. However, this does not mean that the actors are forgetting the success or failure or format of a social construct; the actor can bring it in as prior knowledge to new negotiation processes. Thirdly, the ending can happen when the social construct is not reinforced over time and forgotten; the social construct can still exist, written on paper, but actually the influence of the construct already ended and is not reinforced anymore, or the context changed and the social construct is not adapted to the new situation. Fourthly, the actor suppresses the social construct when the economic or psychological cost of following the social construct is too high.

The processes of creation, evolution and ending depends on the support for the norm (Axelrod, 1986), such as the dynamics of the environment (e.g. entering and exiting actors), regime change and the control process that monitors these processes.

Control and monitoring are processes[28] that determine the actual creation, evolution and ending of a social construct. During creation, actors are involved in controlling the creation of a social construct, e.g. in case of a lunar-registrar, an institution has to control and acknowledge the legitimacy of the start-up. During the evolution process, the social construct and attached norms are due to change, because attached norms may evolve or are violated, e.g. actors can start to re-negotiate about other terms of the social construct in order to modify the social construct.

Therefore, the process of control is vital for determining if the other actor(s)

[27] Retr. 27-03-2006: http://ue.eu.int/ueDocs/cms_Data/docs/pressData/en/misc/89035.pdf

[28] These processes determine other processes, therefore they are often referred to as meta-processes, e.g. meta-norms or in our case meta-social constructs.

are still aware of the conditions and associated norms of a social construct. Conte and Castelfranchi (1995b) state that in a group of addressees to which a norm applies, the norm should be respected by all its addressees; it is a necessary consequence of the *normative equity principle* defining that agents want their 'normative costs' to be no higher than those of other agents subject to the same norms. Actors stick to a norm as long as the advantages of following up a norm outweigh the advantages of not following the norm, i.e. the actor has to comply with the norm and if necessary defend the norm or give up the norm (when the costs of obedience are too high compared to punishment). However, rewards (positive approach) can also be a drive to support norms, e.g. the no-claim discount of insurance companies rewards and enforces people to be more careful about damaging their insured property.

A closely related, non-monetary characteristic that keeps a norm system alive is *reciprocity*. "Reciprocity of norms means that a norm works because there will be a beneficial effect for everyone if everyone behaves according to that norm. A norm tells you how you ought to behave, and you trust that other actors will behave in that way as well" (Gazendam, 2003, p. 17). However, even reciprocity requires some monitoring effort.

The evolution process is always subject to control mechanisms of punishments and rewards, thereby reinforcing norms that lead to certain expected and desired behaviours. In other words, a social construct as part of a community is always subjected to meta-processes that either reinforce, punish or abandon the use of a certain social construct. The social construct itself can become a meta-process for the maintenance of other social constructs when it is a prerequisite for those other constructs. In bureaucratic organisations or institutions, these dependencies can grow to immense proportions; so immense that for instance a minister of justice of a country cannot comprehend it anymore and requires consultants to advise him or her in speaking justice. Hence, the roles and relations in a bureaucratic system form a complex network of dependencies between social constructs in the form of power and authority relations.

We are aware that many systems of social constructs are based on trust, power and relationships between actors. However, in this dissertation we first require a simplified social actor model that is able to (socially) construct simple (coordination) mechanisms first. The implementation and studying of more complex social constructs such as found in bureaucratic organisations is beyond the scope of this dissertation.

In the following section, we close the chapter and discuss the meaning(s) of the social construct and the requirements necessary for the actor to become a social actor. These requirements give guidelines for the modelling—see chapter 5—of a social actor with help of social constructs that enables the actor to be socially situated in its environment and share social constructs with others.

3.5 Discussion

In this chapter we have analysed and determined that social constructs are necessary for an actor to 'behave socially', i.e. social interaction requires certain aspects of actors that enables them to behave socially, cooperate and coordinate

their (joint) activities. With the concept of social construct, we give an answer to the question that we have raised in chapter 1: "What is required for an actor to exhibit (stable) social behaviour?".

We approached the problem first from a philosophical perspective on social constructivism by arguing that actors socially construct the world and experience the organisation as a social construct, i.e. the social construct is an entity that exists because of its mutual acknowledgement by actors that participate in a community. Secondly, we argue that a social construct can be traced back to representations in the mind of the actors in the community (methodological individualism) and as a coordination mechanism structures the interaction among people. Hence, the actor lives in an environment—Umwelt or information field[29]—in which the actor is physically and socially situated and able to create, process, use and transfer signs; a semiotic process or semiosis that enables the actor to construct worlds or representations in their mind (Innenwelt). Thirdly, we argued that semiosis requires a communication process necessary to socially interact with others and that such a communication process presupposes a signification system, a system that enables to interpret/decode and produce/code signs. The last argument we want to make is that such a signification system has to be supported by *cognition* possible in the form of a physical symbol system that manipulates those signs. We argue that such a system is necessary for an actor to be social, i.e. for an actor to be social it has to construct the social world with help of representation of social constructs, thereby creating social representations (or needs) of other actors and the community in which the actor lives.

As mentioned before, the concept of social construct is the contribution of this chapter. We argue that the social construct is a requisite for the cognitive actor—explained in the next chapter—to act socially and coordinate actions with other actors. Throughout the chapter, the social construct is applied at different levels of abstraction. The following figure gives an overview of the meaning of the social construct in a society.

We summarise the four meanings of social construct as follows. First of all, a social construct is the result of interaction between actors. At the social level, actors habituate their action and thereby create a habitat of social constructs. Sign production delivers codes or signs (the semiotic level) to which actors assign (shared) meaning. This shared meaning or knowledge becomes normative at the moment it influences behaviour of the members in the community, e.g. norms that guide behaviour such as the respect for older people. In order to change and understand meaning, actors require a signification system (Eco, 1976). Such a signification system needs to be grounded in the actor, which can be done with help of a plausible cognitive architecture at the cognitive level (functional level), e.g. ACT-R (Anderson & Lebiere, 1998) or RBot, respectively chapter 4 and chapter 5; a cognitive architecture enables the actor to process and

[29]Stamper (1992) proposed a new paradigm [...]: *the information field*...[People]...are all governed by the forces in the information fields and therefore behave accordingly. These forces are related to their interests, assigned functions, tasks, objectives, personal values and organisational goals. These forces may be present in the forms of formal and informal rules, beliefs, cultural habits and conventions, which can be called *norms*. (Liu, 2000, p. 8)

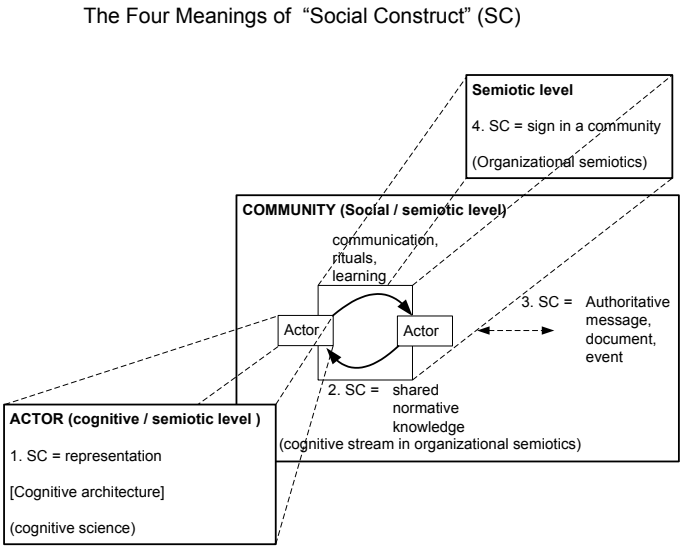


Figure 3.7: Four meanings of a social construct.

produce signs.

The social construct or a derivative of the social construct (sign) needs to be present at the cognitive, social and semiotic level. In the community, the social construct is present as a sign(-structure) (4). These signs or social constructs are exchanged by actors in the community with help of communication and rituals. The exchange is made possible by a medium that allows the transport of social constructs (3) as messages, documents or events[30]. Communication between actors and the reinforcement caused by the frequent use of social constructs will lead to stable habituated patterns of behaviour (e.g. rituals) and shared normative knowledge (2). Besides that, the individual actor needs a physical symbol system (Newell, 1980) that enables it to hold *representations*, process and produce signs/symbols/social constructs (1) and exhibit intelligent action.

Thus, social constructs are representations that are stored in a cognitive architecture—a signification system that interprets and operates on representations (symbols in the mind)—and/or are stored in the environment as for instance artefacts or documents (symbols in the world). The social construct is reproduced or transferred and maintained by social interaction between actors. Because of its internal and external representation, the social construct functions as a glue that connects the actor having a cognitive architecture, via its signification system and communication process with society and culture.

The concluding assumption for an actor in order to exhibit social behaviour is that a social actor requires at least the capability (1) of acting socially in groups

[30]Social construct theory as it has been explained here has been developed within multi-actor theory and the cognitive stream of organisational semiotics and gives, by a theory for the community at the semiotic/social level, a cognitive and semiotic basis to the more general concepts and mechanisms described by sociology and social psychology for the community at the social level.

(organisations), (2) of communicating social constructs to other actors, (3) of storing social constructs, e.g. habits, norms, rules and so on, in the form of representations, (4) of storing knowledge of the world in representations, and (5) a cognitive architecture that supports the maintenance of representations in memory and operates on these representations.

In this chapter we mentioned the cognitive architecture but we did not thoroughly discuss the cognitive architecture. The cognitive architecture has its roots in the field of cognitive science, e.g. philosophy of mind, cognitive psychology and computational theories of cognition. In this research, the cognitive architecture is in the centre of attention. Therefore, the next chapter is devoted to the discussion about cognitive science and a selection of a cognitive architecture.

CHAPTER 4

The Cognitive Actor

THE social actor, as explained in the previous chapter, is concerned with the social construction of the world around him and how this world can be represented in the mind of the actor. In other words, the actor is not only physically, but also socially and culturally embedded (or situated) in the society. On the other hand, the cognitive actor, despite the fact that it is connected with the outside world by systems of perception and action, is more concerned with internal representations and operations on those representations.

We have argued in the previous chapter that the social level can be characterised as social representations, e.g. as social structures, social constructs, and that social representations are symbols in the mind of the cognitive actor that can undergo the same operations as any other symbol or representation in the mind of the actor. In this chapter we focus on cognitive science—especially on the cognitive architecture—and leave the social actor alone. The implementation of the social construct as a representation will be described in the next chapter.

In chapter 1, we asked ourselves what kind of an actor would be capable of handling signs, relations and social constructs. In this chapter, we argue that such an actor should have a cognitive architecture, i.e. be cognitive plausible, in order to handle *representations* such as signs, relations and social constructs. We argue for the need to adopt theories of cognition and cognitive architectures. However, we neither give a complete reconstruction of the theory of cognition, nor present an extensive philosophical and in-depth cognitive discussion on what precisely cognitive architectures and their complex domain of applications are. Instead, we rely on existing theories of cognition about cognitive architectures; theories that have been thoroughly tested and have thereby proven their robustness over the last two decades.

We start with section 4.1 that discusses cognitive science as an interdisciplinary field in which cognitive architectures originate. This encompasses a dis-

cussion on levels of description. We also expose the reader to approaches such as connectionism and embodied (or situated) cognition that present different understandings of how the mind is constructed. These two approaches, even though they are claimed to be significantly different from the classical approach. We argue that their arguments are complementary rather than contradictory arguments to the classical approach. This discussion becomes apparent in our extensions of the cognitive architecture in the next chapter.

We give an overview of the cognitive architecture in section 4.2. Specifically, we consider two cognitive architectures as good candidates for implementing a cognitive plausible actor. In section 4.3, we elaborate the selection of one of these architectures[1]—ACT-R—that best suits our research purposes.

In section 4.4, we detail the internal modules and mechanisms of ACT-R, because its analysis functions as a guideline for the modelling and facilitated comprehension of a new cognitive architecture, i.e. RBot (see chapter 5).

In the final section—section 4.5—we discuss issues concerning ACT-R, and the changes or additions that we make to the principles of ACT-R in order to create a platform of cognitive actors that is suitable for our simulation demonstrated in chapter 6.

4.1 Cognitive science

Cognitive science is a multi-disciplinary field that draws on artificial intelligence, neuroscience, philosophy, psychology, and linguistics. The advocates of each discipline are interested in studying the brain and mind. They see the brain/mind as a complex system that receives, stores, retrieves, transforms, and transmits information (Stillings, Weisler, Chase, Feinstein, Garfield, & Rissland, 1995). Cognitive scientists claim—in opposition to the behaviourists or ecologists—that the mind is more than a simple stimulus-response mechanism, i.e. the mind is a complex construction or architecture that interprets the world and cannot be explained by physical or behavioural laws alone. The presence of *cognitive interpretation* (mechanisms) in the mind of the actor is expressed by Pylyshyn in the following statement:

> The critical aspect of the connection between stimulus conditions and subsequent behavior—the aspect that must be incorporated in the theoretical account if the latter is to capture the systematicity of the person's behavior—is that it is (a) the environment or the antecedent event *as seen or interpreted by the subject*, rather than as described by physics, that is the systematic determiner of actions; and (b) actions performed with certain *intentions*, rather than behaviors as described by an objective natural science such as physics, that enter into behavioral regularities. (Pylyshyn, 1984, p. 9)

Apparently, there is more in the mind than solely a response to signals of the environment. The mind is built up out of representations and representational schemes including "rules for building complex symbolic structures out of

[1]Such a selection process does not reject other cognitive architectures or claim that one is better than the other. It only shows which architecture conforms most to our research purposes.

simpler ones[, which] are called *combinatorial, generative or productive*" (Stillings et al., 1995, p. 5).

Algorithms, assumed to be present in the mind, produce these new symbols out of existing symbols. For instance, an addition of "9" and "8" creates a new symbol "17". However, algorithms are formal operations on symbols and are different from the representational relations between symbols and what they stand for. This difference is in the level of description, a topic that was discussed in both chapters 1, and 2. We made a distinction in several levels of description, from physical to social. In the next section, we continue the discussion about the levels of description, but for the time being leave the emphasis on *the mind of the individual*.

4.1.1 Levels of description

First of all, we want to recall the levels of description of Dennett (1978), discussed in chapter 2. Dennett distinguishes a rational level—a level that contains a "belief-system" in relation to the actor's goals, a functional level—a level with descriptions in the terms of functions of the system, and a physical level—a level that describes the physical properties of functional components.

Following up Dennett, many other cognitive scientists/psychologists have acknowledged that the mind cannot simply be described by a physical or rational level alone. Anderson (1990), see table 4.1, provides a nice comparison between the levels of description assigned by various cognitive scientists. Cognitive scientists are not interested much in the physical (or biological) level, because the details of the physical level are still unclear. Therefore, cognitive scientists depict the physical level as a separate layer, i.e. "the implementation layer is an approximation to the biological level" (Anderson, 1990, p. 18).

<table>
<tr><th>Dennett

(1978)</th><th>Marr

(1982)</th><th>Pylyshyn

(1984)</th><th>Rumelhart & McClelland
(1986)</th><th>Newell

(1982)</th><th>Anderson

(1990)</th></tr>
<tr><td>Intentional Level</td><td>Computational theory</td><td>Semantic Level</td><td></td><td>Knowledge Level</td><td>Rational Level</td></tr>
<tr><td rowspan="2">Functional Level</td><td rowspan="2">Representation and Algorithm</td><td>Algorithm</td><td>Macrotheory/ Rules</td><td>Program Symbol Level</td><td>Algorithm</td></tr>
<tr><td>Functional Architecture</td><td>Microtheory PDP models</td><td>Register Transfer Level</td><td>Implementation</td></tr>
<tr><td>Physical Level</td><td>Hardware implementation</td><td>Biological Level</td><td></td><td>Device</td><td>Biological</td></tr>
</table>

Table 4.1: Levels of description of various cognitive scientists (Anderson, 1990)[2].

Second, we should draw a distinction between the algorithm level and the implementation level, i.e. "[t]here is a distinction to be made between (a) the mental procedures and knowledge we possess that enable use to behave adap-

[2]We have added Dennett (1978) and removed Chomsky (1965) from the original table.

tively, and (b) the mechanisms that implement these procedures and knowledge" (Anderson, 1987, p. 468)[3].

The highest level in cognition is the semantic, intentional, knowledge or rational level, i.e. according to Newell, there should be another level above the symbol or algorithm level:

> *The Knowledge Level Hypothesis*. There exists a distinct computer systems level, lying immediately above the symbol level, which is characterized by knowledge as the medium and the principle of rationality[4] as the law of behavior. (Newell, 1982, p. 99)

The highest level allows one to develop psychological theories. It is not directly concerned with the algorithm or implementation level, or with what the mechanisms of the mind are about.

> Rather, it is about constraints on the behavior of the system in order for that behavior to be optimal. If we assume that cognition is optimized, these behavioral constraints are constraints on the mechanisms. This level of analysis is important, because it can tell us a lot about human behavior and the mechanisms of the mind. ...[I]t leads us to formal models of the environment from which we derive behavior. Thus, its spirit is one which focuses us on what is outside the head rather than what is inside and one which demands mathematical precision. (Anderson, 1990, pp. 22–23)

An example of a rational analysis is known from Dennett (1987)'s intentional level that ascribes beliefs, desires and intentions to actors:

> ...first you decide to treat the object whose behavior is to be predicted as a rational agent; then you figure out what beliefs that agent ought to have, given its place in the world and its purpose. Then you figure out what desires it ought to have, on the same considerations, and finally you predict that this rational agent will act to further its goals in the light of its beliefs. A little practical reasoning from the chosen set of beliefs and desires will in many—but not all—instances yield a decision about what the agent ought to do; this is what you predict the agent *will* do. (p. 17)

According to Anderson (1990), the rational analysis has some advantages. In the first place, a rational approach has the advantage of avoiding complications of the mechanistic approach.

> [First o]ne has a theory that depends on the structure of an observable environment and not on the unobservable structure of the mind...[Second, i]t offers explanation for why the mechanisms compute the way they do...[, and third, i]t offers real guidance to theory construction...[instead of relying]...on very weak methods to search a very large space of psychological hypotheses. (p. 30)

[3]This is similar to Pylyshyn (1984)'s distinction in Algorithm and Functional Architecture.

[4]For a definition of the principle of rationality, see 4.3.1: *cognitive principles*.

In studying Multi-Agent Systems (MAS), often the rational or intentional level is considered well-suited for describing (behavioural) systems, e.g. the game theory applied in economics or the rational (logical) agent (cf. Wooldridge, 2000) in MAS. In this dissertation, we pay attention to the intentional level, because we incorporate goals and interaction with the environment/social world as well. However, we also want to incorporate the functional level—algorithm and implementation levels—besides the rational or intentional level. In other words, an actor should contain (1) *representations*, (2) *cognitive mechanisms or algorithms*, and (3) *functional mechanisms*[5], because we argue that human behaviour cannot solely be explained by simplified stimulus-response behaviour. A design or a model in the form of a *cognitive architecture* is a proven methodology in unfolding the essentials of human behaviour in a more accurate way than that provided by the behaviourist approach.

The cognitive architecture is based on the fundamental view that in cognitive science an intelligent system is not completely homogeneous, i.e. it must consist of a number of functional subsystems, or modules, that cooperate to achieve intelligent information processing and behaviour. This view is supported by empirical results in cognitive psychology (Stillings et al., 1995).

This chapter primarily focuses on the classical approach of cognition. However there are also two other areas—*connectionism* and *embodied cognition*—within cognitive science that deserve attention. For instance, ACT-R, as discussed in this chapter, applies connectionism-like assumptions in its associative memory. In the case of embodied cognition, we adopt the ideas of the subsumption (architecture) of behaviours for constructing the awareness of social situations.

4.1.2 Connectionism and embodied cognition

We will explain shortly what the *connectionist approach* means as regards the construction of a cognitive architecture. The connectionist approach originates from the PDP[6] group (Rumelhart et al., 1986) and presents a quite different approach from that of the classical cognitive approach; it even seeks to replace it.

A connectionist architecture consists of a network of computationally simple processing *units* or *nodes* that are interconnected with each other and are capable of transporting only simple *signals*[7]. A short description of connectionism is given by Fodor and Pylyshyn (1988, p. 5):

> Connectionist systems are networks consisting of very large numbers of simple but highly interconnected "units". Certain assumptions are generally made both about the units and the connections: Each unit is assumed to receive real-valued activity (either excitatory or inhibitory or both) along its input lines. Typically the units do little more than sum this activity and change their state as

[5]These three requirements are present at respectively, the rational (knowledge) level, the algorithm level, and the implementation level (or functional architecture).

[6]Parallel Distributed Processing

[7]We refer to the previous chapter for the definition of *signals* and how they differ from *signs*.

> a function (usually a threshold function) of this sum. Each connection is allowed to modulate the activity it transmits as a function of an intrinsic (but modifiable) property called its "weight". Hence the activity on an input line is typically some non-linear function of the state of activity of its sources. The behavior of the network as a whole is a function of the initial state of activation of the units and of the weights on its connections, which serve as its only form of memory.

The attraction to connectionist networks is due to their similarity to neural networks. Neurons have several inputs that can trigger a neuron to fire, similar to the threshold level of a unit of a connectionist model; connections between the units tend to have a structure that is similar to the structure of connections between axons and dendrites. However, in order for a connectionist model to constitute a theory of a cognitive architecture, it has to make claims about *representations* and processes that occur at the semantic or knowledge level. Symbolic computation is something that is lacking in a connectionist model[8]. For instance, it is difficult for connectionist models to reason about the difference between 'John loves Mary' and 'Mary loves John'. According to classical cognitive scientists, it can be argued that connectionist models are suitable for lower levels, such as the latency of production firing and retrieval of information from memory.

We give attention to connectionism here, because in hybrid architectures, like ACT-R, activation as a property of *symbolic chunks* is similar to activation of a unit in connectionist models. A connectionist model has the ability to spread activation over time. In other words, the activation of a unit is spread over other units that are connected to it. This also occurs in classical models that borrow properties of connectionist models.

Embodied Cognition[9] (EC), part of the so-called New AI, is an opponent of GOFAI stating that an actor is not dependent on internal representations, but is a more or less behavioural system situated in and responding to its environment:

> Instead of emphasizing formal operations on abstract symbols, [EC] focuses attention on the fact that most real-world thinking occurs in very particular (and often very complex) environments, is employed for very practical ends, and exploits the possibility of interaction with and manipulation of external props. It thereby foregrounds the fact that cognition is a highly *embodied* or *situated activity* and suggests that thinking beings ought therefore be considered first and foremost as acting beings. (Anderson, 2003, p. 91)

Rodney Brooks has had the most impact in the field of New AI. He published a collection of papers in his book *Cambrian Intelligence* (Brooks, 1999) that covers many aspects of EC and the field of robotics. In one of those papers, "Intelligence without Representation", Brooks takes a radical position against classical AI (Brooks, 1991b, p. 140):

[8] Fodor and Pylyshyn (1988) provide strong arguments as to why connectionist models cannot replace classical models and that most of them concern implementation issues.

[9] Within Cognitive Science, Embodied Cognition falls in the category of New AI, into which we can classify Embodied Action, Situated Cognition or Situated Action as well.

> We have reached an unexpected conclusion (C) and have a rather radical hypothesis (H).
>
> **(C)** When we examine very simple level intelligence we find that explicit representations and models of the world simply get in the way. It turns out to be better to use the world as its own model.
>
> **(H)** Representation is the wrong unit of abstraction in building the bulkiest parts of intelligent systems.
>
> Representation has been the central issue in artificial intelligence work over the last 15 years only because it has provided an interface between otherwise isolated models and conference papers.

Similar to connectionism, this statement attacks *the existence of representations* in a manner equivalent to the debate that the connectionists have postulated against representations. However, this attack is from the outside rather than the inside, i.e. representations prevail in the world and not in the mind.

The counterclaim of classical AI is that an intelligent organism cannot reason without representations or a physical symbol system. Kirsh (as cited by Anderson, 2003) states that:

> ...[I]f a task requires knowledge about the world that must be obtained by reasoning or by recall, rather than by perception, it cannot be classified as situation determined. Principle candidates for such tasks are:
>
> - Activities which involve other agents, since these often will require making *predictions* of their behaviour.
> - Activities which require response to events and actions beyond the creature's current sensory limits, such as taking precautions now for the future, avoiding future dangers, contingencies, idle wandering—the standard motive for internal lookahead.
> - Activities which require understanding a situation from an objective perspective such as when a new recipe is followed, advice is assimilated, a strategy from one context is generalized or adapted to the current situation. All these require some measure of conceptualization.
> - Activities which require some amount of problem solving, such as when we wrap a package and have to determine how many sheets of paper to use, or when we rearrange items in a refrigerator to make room for a new pot.
> - Activities which are creative and hence stimulus free, such as much of language use, musical performance, mime, self-amusement.
>
> These activities are not isolated episodes in a normal human life. Admittedly, they are all based on an underlying set of reliable control

> systems; but these control systems are not sufficient themselves to organize the *global* structure of the activity. (Kirsh, 1991, pp. 173–174)

Although we also think that Brooks' hypothesis and conclusion are too strong, we agree with the following key aspects that characterise his *behaviour-based* style of work (Brooks, 1991a, 1999, pp. 138–139):

Situatedness The [actors] are situated in the world—they do not deal with abstract descriptions, but with the here and now of the world directly influencing the behavior of the system.

Embodiment The [actors] have bodies and experience the world directly—their actions are part of a dynamic with the world and have immediate feedback on their own sensations.

Intelligence They are observed to be intelligent—but the source of intelligence is not limited to just the computational engine. It also comes from the situation in the world, the signal transformations within the sensors, and the physical coupling of the [actor] with the world.

Emergence The intelligence of the system emerges from the system's interactions with the world and from sometimes indirect interactions between its components—it is sometimes hard to point to one event or place within the system and say that is why some external action was manifested.

These aspects are in line with the approach taken by Brooks to decompose an actor control problem. This is carried out in task achieving behaviours such as avoiding objects, wandering, exploring, building maps, monitoring changes etc. rather than in *functional units* such as perception, modelling, planning, task execution, and motor control. The idea of Brooks was to design a completely autonomous mobile robot that would perform many complex information processing tasks in real time. This lead to the *subsumption architecture*, see figure 4.1, that broke down an actor into vertical modules, each addressing a small part of the total behaviour (Brooks, 1986).

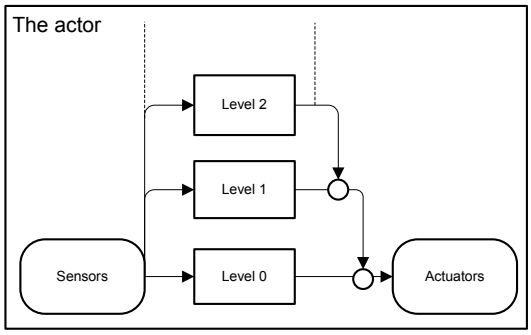


Figure 4.1: The subsumption architecture (Brooks, 1986).

The subsumption architecture is constructed in the following way. As a first step, you build a system at level 0 that is stable, and assign it as *level 0 competence*. Another layer that is able to influence behaviour at the lower level, called the first level control system, is added to this layer. In this way many levels of behaviour can be built on top of each other, which produce a complex system of (hard-wired) behaviours.

For example, assume a robot on Mars[10] that has the main goal of creating a map of the surface (competence level 2) with help of exploring that is done by looking for places that are reachable (competence level 1). And finally, the lowest level in the system (level 0 competence) makes sure that the robot survives by evading objects. The organisation of levels allows the lower levels to overrule or inhibit the higher levels. In this way, the lower levels act as fast-responding mechanisms (reflexes) while the higher levels determine the direction where to go and try to fulfil the overall (main) goal.

In the next chapter, we integrate embodied or situated cognition, and adopt the principles of the *subsumption* architecture. We argue not only that the subsumption architecture is suitable for detecting physical changes in the environment and therefore creates the notion of *physical* situatedness, but that the same principles can be applied to creating an actor that responds to *social* changes (changing social constructs) and thereby is *socially* situated. We close the discussion about embodied cognition here with the claim that a cognitive architecture should have representations and a physical symbol system, and that embodied or situated cognition can support the symbolic cognitive architecture in responding better to changes in the environment.

In the following sections, we focus our discussion on the cognitive architecture and the selection of such an architecture. The selected architecture will form the core of our actor—RBot—that is discussed in the next chapter.

4.2 The Cognitive Architecture

In the previous chapter we argued that an organisation consists of a collection of actors that have their own mental representation of the organisation as a whole. Such actors are in need of a cognitive model that supports representations and operations on those representations. A cognitive architecture implements such a model, and gives the actors the ability to maintain beliefs, desires and intentions (goals) at the rational level, and to manipulate representations at the algorithm level. They also have functional mechanisms (e.g. memory storage and retrieval functions) at the implementation level, a level that is an approximation of the physical level.

Taatgen (1999, p. 28) defines an architecture of cognition as an algorithm that simulates a non-linear theory of cognition. This algorithm can be used to make predictions in specific domains and for specific tasks, see figure 4.2. Hence, the simulation of cognitive phenomena consists of, apart from the architecture, a specific task domain (or environment) that needs to be implemented, combined

[10]For the interested reader, see also Brooks' paper "Fast, cheap and out of control: a robot invasion of the solar system" (Brooks & Flynn, 1989).

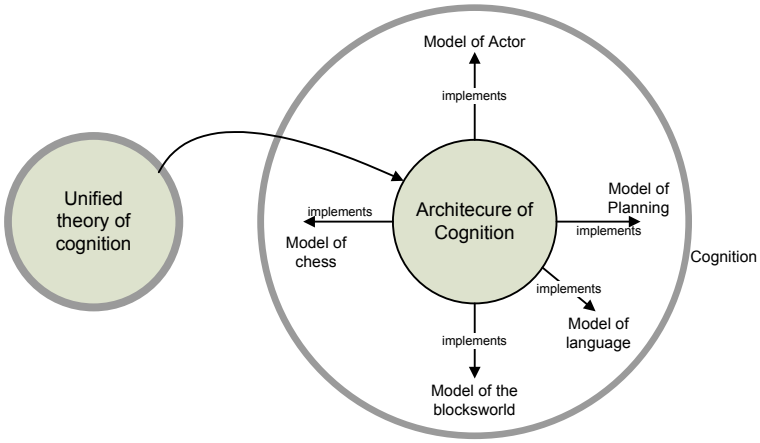


Figure 4.2: Relationship between theory, architecture, models, actor and cognition (adapted from Taatgen, 1999).

with knowledge necessary to complete the tasks in that domain.

A *cognitive architecture* is a theory about the computational structure of cognition (Anderson & Lebiere, 1998; Newell, 1990; Pylyshyn, 1984), i.e. it specifies a set of processes, memories, and control structures that are capable of encoding content and executing programs (Lewis, 2001). The two mainstream cognitive architectures[11] in artificial intelligence are SOAR (Laird et al., 1987) and ACT-R (Anderson & Lebiere, 1998). A choice between one of these architectures seems to be the most economical and logical one for our cognitive actor[12].

4.2.1 SOAR and ACT-R

In this section, we give a short summary of the cognitive architectures SOAR and ACT-R. Some details of the cognitive plausible actor were already introduced in chapter 2. These encompassed the components or properties of perception, (working) memory, cognition, learning, action, motor-control commonly found in cognitive architectures, like SOAR and ACT-R.

4.2.1.1 Summary of SOAR

The implementation of a cognitive architecture started with Newell and Simon (1972) who introduced a production-system theory of human cognition. Newell tested a number of different production systems, concluding with his SOAR theory of human cognition (Newell, 1990). The SOAR theory posits that cognitive behaviour has at least the following characteristics: it (1) is goal oriented; (2) takes place in a rich, complex, detailed environment; (3) requires large

[11] EPIC (Meyer & Kieras, 1997) and 3CAPS (Just & Carpenter, 1992) are examples of other cognitive architectures not discussed here. For a discussion, see Taatgen (1999).

[12] Although the cognitive actor is part of a multi-actor system, the emphasis here is not on MAS but on the cognitive architecture of the actor.

amounts of knowledge, the use of symbols and abstractions; (4) is flexible, and a function of the environment; and (5) requires learning from the environment and experience (Lehman et al., 2006). The theory and properties of cognitive behaviour have been the requirements for developing the cognitive model or architecture SOAR.

SOAR[13] (**S**tate, **O**perator **A**nd **R**esult) is a *cognitive* or *problem solving architecture* in which problem solving occurs with the help of tasks accomplished in problem spaces, i.e. the *problem space hypothesis*[14]. SOAR solves tasks in the following way:

> To realize a task as search in a problem space requires a fixed set of *task-implementation* functions, involving the retrieval or generation of: (1) problem spaces, (2) problem space operators, (3) an initial state representing the current situation, and (4) new states that result form applying operators to existing states. To control the search requires a fixed set of *search-control* functions, involving the selection of: (1) a problem space, (2) a state from those directly available, and (3) an operator to apply to the state. Together, the task-implementation and search-control functions are sufficient for problem space search to occur. (Laird et al., 1987, p. 11)

SOAR has an architecture—see figure 4.3—that consists of five main parts:

1. A *Working Memory* that holds the complete processing state for problem solving in SOAR and exists out of three components: (1) a context stack that specifies the hierarchy of active goals, problem spaces, states and operators; (2) objects, such as goals and states; and (3) preferences that encode the procedural search-control knowledge, i.e. structures that indicate the acceptability and desirability of objects.

2. A *Decision Procedure* that examines the preferences and the context stack, and changes the context stack, i.e. it influences what goal or action is put on top of the stack.

3. A *Working Memory Manager* that maintains and cleans up the working memory from elements that are not needed anymore.

4. A *Production Memory* which is a set of productions that can examine any part of working memory, add new objects and preferences, and augment existing objects, but cannot modify the context stack.

5. A *Chunking Mechanism* is a learning scheme for organising and remembering ongoing experience automatically. New chunks are created, when

[13] A comprehensive and detailed explanation of SOAR can be found in (Laird et al., 1987; Lehman et al., 2006; Lewis, 2001; Newell, 1990) or at http://sitemaker.umich.edu/soar.

[14] "The rational activity in which people engage to solve a problem can be described in terms of (1) a set of states of knowledge, (2) operators for changing one state into another, (3) constraints on applying operators and (4) control knowledge for deciding which operator to apply next" (Newell, 1990, p. 33)

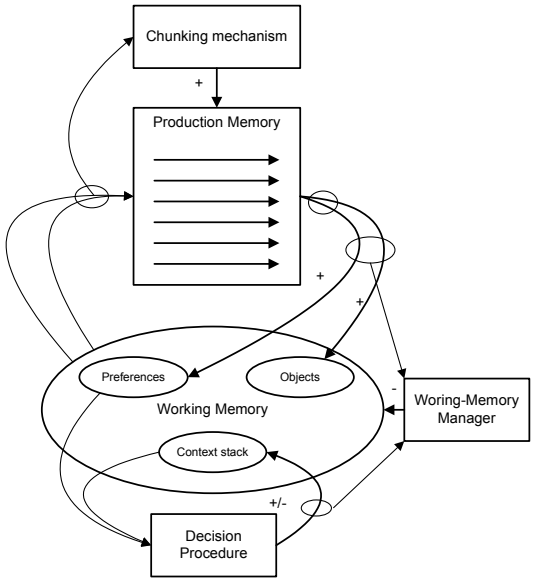


Figure 4.3: Architectural structure of SOAR (Laird et al., 1987).

SOAR encounters an insolvable problem or *impasse*. The impasse causes SOAR to create a subgoal, which is created to find the necessary knowledge. The chunking mechanism stores away the knowledge so that under similar circumstances in the future, the knowledge will be available.

The complex mechanism behind SOAR works as follows: (1) a certain problem is projected onto a problem space; (2) the representation of the problem and its context objects is changed by adapting the current problem space to a new one; (3) the replacement is controlled by the *decision cycle* that consists of (a) the *elaboration phase*—new objects, new augmentations of old objects, and preferences that are added to working memory, and (b) the *decision procedure* that either replaces an existing object or creates a subgoal in response to an impasse upon which the next cycle starts. The general behaviour of SOAR is a divide-and-conquer approach oriented towards a goal, i.e. during its search—with the help of the decision cycle(s)—it will make use of subgoals to solve sub-problem spaces that will contribute to solving the main goal.

4.2.1.2 Summary of ACT-R

The ACT-R (**A**daptive **C**ontrol of **T**hought, **R**ational) architecture (Anderson & Lebiere, 1998) is a production system similar to SOAR, but with many differences regarding its implementation. ACT-R (Anderson, 1990; Anderson & Lebiere, 1998; Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004) consists of a set of modules: perception and motor (visual and manual module) that connect ACT-R with the outside world, a declarative memory retrieving information from memory, a memory containing productions, and a goal module

for keeping track of current goals and intentions. The coordination and behaviour of these modules is controlled by a production system, similar to SOAR. In addition, ACT-R has a set of buffers in between the modules and the production system. The contents of the buffers is regulated by separate processes that are active in the modules, i.e. there is (some) independency caused by separate systems that determine the contents of the buffer. For instance, the contents of the visual buffer represent the products of complex processes of the visual perception and attention systems. Furthermore, the decision or "critical" cycle in ACT-R is similar to that in SOAR: "the buffers hold representations determined by the external world and internal modules, patterns in these buffers are recognized, a production fires, and the buffers are then updated for another cycle" (Anderson et al., 2004, p. 1038).

ACT-R's distinction from SOAR lies not so much in its processing structure at the algorithm level, but at the implementation level or in its functional architecture. Learning does not only unfold, through the creation of new chunks[15] by solving (sub)goals, as that in SOAR, but occurs also at the sub-symbolic level. Hence, learning occurs not only by analogies and creation of new chunks as the result of production firing in the decision cycle, but also at the sub-symbolic level of productions and chunks. Every production and chunk maintains an activation level that increases with successful use, which leads to faster retrieval of chunks and production matching and execution. The combination of sub-symbolic activation with symbolic chunks makes ACT-R a hybrid architecture[16].

4.3 Comparing SOAR and ACT-R: selecting a cognitive architecture

In this section, we compare the properties or characteristics of both architectures for similarities, followed by a more in-depth discussion of the differences to inform our selection of one of the architectures[17].

First, we look at a set of constraints. Newell (1980, 1990) described the mind as a solution to a set of functional constraints (e.g. exhibit adaptive (goal-oriented) behaviour, use of language, operate with a body of many degrees of freedom) and a set of constraints on construction (a neural system growing by evolutionary processes, arising through evolution) (Lewis, 2001).

Both architectures, SOAR and ACT-R, conform to three functional constraints a) exhibiting flexible, goal-driven behaviour, b) learning continuously from experience, and c) exhibiting real-time cognition. However, ACT-R also conforms to the constraint on construction by implementing sub-symbolic properties that exhibit a connectionist-like neural network.

Second, in line with Lewis (2001), SOAR and ACT-R both implement the five major technical principles in cognitive science:

[15] Chunks in SOAR are productions, while in ACT-R they are facts or declarative chunks.

[16] Hybrid architectures are a mix of connectionism (neuron-like networks) and symbolism (physical symbol system).

[17] The comparison and selection are mainly based on ACT-R version 4.0 and the SOAR version without episodic and semantic memories.

1. The *physical symbol system hypothesis* that assumes that only physical symbol systems support intelligent, flexible behaviour. The system is required to process, manipulate and compose *symbols and symbol-structures* to support intelligent, flexible behaviour.

2. A *cognitive architecture*: a fixed structure that processes the variable content and executes programs specified by a set of processes, memories and control structures. Different cognitive models can be developed for a specific task environment (problem-space) with the help of a single architecture. What is of importance is the structure of the architecture and the cognitive model's program and *not* the implementation in a programming language. In other words, theoretical claims are made about the model and not about the underlying programming language.

3. A *production system*, consisting of productions (condition-action pairs) that are matched to patterns in the declarative or working memory, which in turn create state changes in the memory. The behaviour of the system is *goal-oriented*, i.e. the main goal is solved by creating subgoals (divide-and-conquer) when a goal cannot immediately be solved.

4. Cognition in both SOAR and ACT-R is achieved by a search in problem spaces supported by a three stage (recognise, decide, act) automatic/deliberate control structure. The difference between both architectures is that SOAR fires all procedures in parallel[18] and matches those to the current state or goal[19]—resulting in a collection of (different) operators and end-states, after which SOAR decides which state is consistent and acts based on that state. On the other hand, ACT-R first selects only the procedures that match with the current goal, selects between those matching procedures (conflict resolution), and fires/enacts only the selected procedure which results in state change(s). Both architectures have a so-called impasse (SOAR) or conflict resolution mechanism (ACT-R) that solves conflicts for SOAR when it is not clear what needs to be done next, i.e. when there are several operators and end-states that are all consistent at the same time, and for ACT-R when there are two or more productions matching at the same time. In the case of SOAR, a new subgoal is created by a decision-procedure, while in ACT-R, the conflict resolution forces the selection of a single procedure based on the success ratio built up from experience in the past.

5. SOAR and ACT-R both have learning processes. SOAR learns by generating new productions in memory by preserving the results of previous problem solving that occurs in response to an impasse. ACT-R, on the other hand, learns not only by creating new declarative chunks in mem-

[18] Procedure and production are used interchangeably as well as in most literature, but they refer to the same phenomena.

[19] The same for state and goal; they are interchangeable, in so far that goal and state are both reacting for executing an action.

ory, but also by production compilation[20] that creates new productions by compiling facts from declarative memory into new productions. SOAR, as well as ACT-R, create faster execution of productions, by recording previous successful paths of solutions. Apart from this, ACT-R applies subsymbolic learning properties to chunks and procedures that give it a finer grain than SOAR. This application also gives ACT-R's architecture the features of a neural network.

In order to select a cognitive architecture, specific selection criteria need to be met. Over and above the constraints and the technical principles, the architectures need to be capable of implementing (or serve as a plug-in into) a multi-actor or computational organisation model based on cognitive plausible actors that are able to coordinate and cooperate to fulfil organisational goals.

4.3.1 Selection of a cognitive architecture[21]

In our research, the selection process requires criteria that determine whether a cognitive architecture is suitable for implementing a computational organisation model. The cognitive architecture not only has to fulfil (a) the cognitive demands to create a cognitive plausible actor, but also (b) the social and organisational demands that are needed to support organisational behaviour.

First, we will address the cognitive principles and the differences between SOAR and ACT-R. Then we will introduce the requirements of a cognitive plausible actor for it to conform to the concept of bounded rationality: "the concept that Simon (1945, p. 80) coined to conceptualize the limitations of "perfect" representations of the environment by the human actor and also of the mental processing capacity of the actor" (Helmhout et al., 2004, p. 9). The last requirement concerns the social and organisational demands. We will adopt requirements from computational mathematical organisation theory: the ACTS theory to address these demands (Carley & Newell, 1994).

Cognitive Principles

According to Anderson (1990), a cognitive architecture, such as SOAR and ACT-R, is defined along the following principles:

1. *Cognitive Impenetrability* (Pylyshyn, 1984; Anderson, 1990, p. 11): "The operations at the functional architecture level are not affected by the organism's goals and beliefs".

   > Although the architecture might vary as a function of physical or biochemical conditions, it should not vary directly or in logically coherent or in rule-governed ways with changes in the

[20] In SOAR, there is only a production memory. In ACT-R there is also a declarative memory: ACT-R creates new chunks in declarative memory; the creation of new productions (production-compilation (Taatgen, 1999)) is present in the newer version ACT-R 6.0.

[21] The selection of the architecture was made based on the characteristics of SOAR and ACT-R in the year 2003.

> content of the organism's goals and beliefs. If the functional architecture were to change in ways requiring a cognitive rule-governed explanation, the architecture could no longer serve as the basis for explaining how changes in rules and representations produce changes in behavior. Consequently, the input-output behavior of the hypothesized, primitive operations of the functional architecture must not depend in certain, specific ways on goals and beliefs, hence, on conditions which, there is independent reason to think, change the organism's goals and beliefs; the behavior must be what I refer to as *cognitively impenetrable*. (Pylyshyn, 1984, p. 114)

For instance, in ACT-R, the sub-symbolic level is not logically coherent or rule-regulated, i.e. only the symbolic level is influenced by the semantic contents of our knowledge (Anderson, 1990).

2. *Physical Symbol System Hypothesis* (Newell, 1980, p. 170):

   > The necessary and sufficient condition for a physical system to exhibit general intelligent action is that it be a physical symbol system.
   >
   > **Necessary** means that any physical system that exhibits general intelligence will be an instance of a physical symbol system.
   >
   > **Sufficient** means that any physical symbol system can be organized further to exhibit general intelligent action.
   >
   > **General intelligent action** means the same scope of intelligence seen in human action: that in real situations behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some physical limits.

3. *Principle of Rationality* (Newell, 1982): "If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action" (p. 102). According to Newell, this principle is part of the Knowledge Level Hypothesis, thereby stating that the "principle of rationality does not have built into it any notion of optimal or best, only the automatic connection of actions to goals according to knowledge" (Newell, 1982, p. 102). In a similar vein, Anderson (1990) states: "there are three terms related by the principle of rationality—goals, knowledge, and behavior. Given any two, one can infer or predict the third. Thus, if we know a person's goals and observe his behavior, we can infer his knowledge" (p. 15).

4. *General Principle of Rationality* (Anderson, 1990, p. 28): "The cognitive system operates at all times to optimize the adaptation of the behavior of the organism". Anderson avoids the use of knowledge and defines a more abstract principle of rationality, by stating that the cognitive system is optimising the adaptation of the behaviour at all times, i.e. the system is continuously trying to be in coherence with itself and with its environment.

> [A] principle of rationality should be defined in terms of a person's experience and not knowledge. The implications of experience are uncertain and fundamentally probabilistic, whereas the implications of knowledge are certain. Because it is experience that people have direct access to and not knowledge, our analysis of what behavior is rational has to face up to the probabilistic structure of uncertainty. (Anderson, 1990, p. 17)

Differences between SOAR and ACT-R

The first *difference* between SOAR and ACT-R is that SOAR operates more at the knowledge level (principle of rationality), while ACT-R operates at the level of optimising the adaptation of the behaviour (also at the sub-symbolic level). Hence, there is a difference in the *theoretical assumptions* between SOAR and ACT-R. The second difference is the *sequential control* of both architectures. Although both architectures do focus on a single goal, the procedures in SOAR can fire in parallel, while the procedures of ACT-R fire in serial. Third, there is a difference in the structure of *long-term memory*: ACT-R distinguishes facts (declarative memory) from productions (production memory) while SOAR only maintains productions in long-term memory. Furthermore, the *learning capabilities* of SOAR are based on two learning mechanisms[22] 'chunking' and reinforcement learning (the adjustment of values of operators). On the other hand, ACT-R has learning at the sub-symbolic level, production compilation (ACT-R version 6.0), history of experience of procedures, and creation of declarative facts. And finally, the *latency derivation* (the time it costs to retrieve and execute) is implemented in a different way for both architectures. While for SOAR there is a constant latency, ACT-R calculates latency based on activation of the needed elements in memory. Table 4.2 gives an overview of the differences between SOAR and ACT-R.

Bounded rationality

Another important criterion for accepting an architecture is the 'bounded rational' nature of the model. This is a term used to designate rational choice that takes into account ecological and cognitive limitations of both knowledge and computational capacity. This relates to the question of how an organism or cognitive architecture makes inferences about unknown aspects of the environment. Gigerenzer and Goldstein (1996, p. 650) state that there are two views con-

[22]The most recent developments show that SOAR has now four architectural learning mechanisms: (1) Chunking automatically creates new rules in LTM whenever results are generated from an impasse. It speeds up performance and moves more deliberate knowledge retrieved in a substate up to a state where it can be used reactively. The new rules map the relevant pre-impasse WM elements into WM changes that prevent an impasse in similar future situations. (2) Reinforcement learning adjusts the values of preferences for operators. (3) Episodic learning stores a history of experiences, while (4) semantic learning captures more abstract declarative statements (Lehman et al., 2006). Apparently architectures are always 'under construction'. For instance, there is a major change in the latest release of SOAR: the introduction of an episodic and a semantic memory, i.e. the latest version of SOAR assumes the existence of different types of memory besides the production memory as well.

ACT-R	SOAR
Foundational assumptions	
- Mechanisms should be implemented in neural-like computation - Cognition is adapted to the structure of the environment (Anderson, 1990) - General principle of rationality	- Humans approximate knowledge level systems - Humans are symbol systems - Principle of rationality
Sequential control	
- Single goal hierarchy - Goals are specific knowledge chunks - Adaptive, satisficing conflict resolution on production	- Single goal hierarchy - Goals are states created by architecture - Knowledge-based, least commitment conflict resolution on operator
Long Term Memory (LTM)	
- Distinctive buffers: declarative(facts) and procedural memory(productions) - Declarative memory is a network with activations and associative strengths	- Uniform LTM: all LTM in procedural form - Graph structure storing procedures
Learning	
- The strength of declarative and procedural memory increases as a power function of practice and decrease as a power function of delay - Procedural knowledge is tuned, based on experience - Procedural knowledge is acquired by production compilation (version 6.0) - Declarative knowledge is acquired through internal production firing and external perception - Activations and associative strengths of declarative memory are acquired through experience	- All long-term knowledge arises through chunking - Reinforcement learning - New procedures are created with the help of chunking
Latency derivation	
- Latencies are derived from the sum of times for each chunk to be matched to each condition in the rule. - Depends on the strength of a rule and the strength of the memory elements it matches	- Constant latency is per recognize-decide-act cycle - Total latency depends on number of decision cycles plus the time for external actions

Table 4.2: Comparison of SOAR and ACT-R (Johnson, 1997).

cerning this question. The first is the "...classical view that the laws of human inference are the laws of probability and statistics...Indeed, the...probabilists derived the laws of probability from what they believed to be the laws of human reasoning (Daston, 1988)". The other view "...concluded that human inference is systematically biased and error prone, suggesting that the laws of inference are quick-and-dirty heuristics and not the laws of probability (Kahneman, Slovic, & Tversky, 1982)". According to Gigerenzer and Goldstein, there is a third way of looking at inference that focuses on "the psychological and ecological rather than on logic and probability theory" (p. 651). This view is the proposal of bounded rationality models instead of the classical rationality models.

Simon (1945, 1956, 1982) stated that human beings are boundedly rational, and that they rather 'satisfice' than optimise. The fundamental characteristics of bounded rationality are the following, which can be incorporated into a cognitive architecture:

1. Limitation on the organism's ability to plan long behaviour sequences, a limitation imposed by the bounded cognitive ability of the organism as well as the complexity of the environment in which it operates.

   Both SOAR and ACT-R work with temporal mappings of declarative knowledge and productions that are active in the current state of the actor. Only a certain set of operators or productions that are active (above a threshold) can be matched with the current state or goal, limiting the actor's possibilities. At the same time, no deterministic mathematics can be applied to identify the next state, or the many that follow after that state. ACT-R adds even more to bounded rationality, i.e. activation levels can drop below the threshold value resulting in 'forgetting' or memory decay.

2. The tendency to set aspiration levels for each of the multiple goals that the organism faces.

   Both SOAR and ACT-R set preferences to expected end-states or procedures (utilities of operators or procedures) that match the current state rather than to goals. Both SOAR and ACT-R lack an explicit intentional module that enables deliberation over goals at the *rational* level.

3. The tendency to operate on goals sequentially rather than simultaneously because of a "bottleneck of short-term memory".

   This argument holds for both architectures. SOAR initially operates as a parallel architecture, because all possible productions fire in parallel, while ACT-R only fires the selected procedure. However, in due course, SOAR selects (possible by impasse) the preferred end-state, ending up with the outcome of one procedure. This *recognize-decide-act cycle* gives SOAR and ACT-R their serial sequence and causes for both a serial/cognitive bottleneck when handling goals (cf. Lehman et al., 2006; Taatgen, 2005; Taatgen & Anderson, 2006).

4. 'Satisficing' rather than optimising search behaviour.

   The actor creates behaviour that 'satisfices' its own preferences, which is most of the time not the optimal economical behaviour to solve solutions for itself or the environment (including other actors). This is due to (1) the *non-deterministic* nature of both architectures, (2) the incomplete and imperfect mental representation of the surroundings, (3) the learning based on actor's own experiences and, (4) particularly in the *case of ACT-R*, lowering activations of stored knowledge (associative memory at the sub-symbolic level) and strengthening it on retrieval.

Social and organisational demands

The cognitive aspects of an architecture discussed up to this point are important for verifying the cognitive plausibility of the architecture. The previous chapter had highlighted the need for an actor to exhibit social stable behaviour in the form of, for instance, communicating with others, maintaining representations of other actors and coordinating tasks and actions with others. The ACTS theory[23] (Carley & Prietula, 1994b) addresses the issue of (shared) tasks and social situatedness by creating an extension of the original model of bounded rationality by

> ... (1) replacing [or extending] the general principles of the boundedly rational agent with a full model of the cognitive agent that exhibits general intelligence; and (2) replacing general notions of environmental constraints with detailed models of two environmental components—the task and the organizational social-situation within which the task and agent are situated. (p. 55)

In SOAR and ACT-R, the modelling of a task (problem-space) is done with the help of goal-directed behaviour, but the organisational setting requires SOAR and ACT-R to understand the (shared) task and social environment to constitute an 'organisational' cognitive architecture. The task and the social axiom of ACTS theory define that such an organisational agent performs one or more tasks in an organisation to achieve specific personal, task and/or social (group and organisational) goals, and that it occupies a (formal and informal) position in the organisation that is a socio-cultural-historical artefact involving one (or more) socially-situated roles. Such an organisational setting can only be created when a cognitive plausible actor is part of a task environment or multi-actor system.

In a MAS, a cognitive actor needs the capability to communicate and exchange instructions (speech) with other actors. Therefore, it is necessary to have a perception and motor module for each actor. The perception and motor define the border for the cognitive architecture. Beyond the modelling of the individual cognition by drawing on cognitive science, one needs to consider, as explained in chapter 3, arguments within social psychology, sociology and organisational theory.

From among existing implementations, SOAR is the only architecture that has some applications in multi-actor systems, like Plural-SOAR (Carley, Kjaer-Hansen, Newell, & Prietula, 1991) or TASCCS (Verhagen & Masuch, 1994). Therefore, it promises to be a good candidate for modelling organisational behaviour.

However, ACT-R cannot be easily outranked despite the fact that it does not have many implementations of multi-actor simulations. It, nonetheless, has a perception and motor module making communication with the outside world feasible. Therefore, it is possible to create a multi-actor simulation without much effort. In addition, ACT-R's learning capabilities, sub-symbolic level properties

[23] ACTS theory: "... organizations are viewed as collections of intelligent Agents who are Cognitively restricted, Task oriented, and Socially situated." (Carley & Prietula, 1994b, p. 55).

and the empirically tested and proven internal workings of ACT-R pose it as a strong candidate as well.

Essential to both architectures is that they can be implemented as multi-actor systems. They have been verified as appropriate candidates for cognitive psychological experiments, most of which are based on interaction with the outside world (requiring perception and motor). It is this capability of interaction with the outside world that makes both architectures suffice in creating multi-actor systems. However, ACT-R with its sub-symbolic properties and empirical validation comes closer to the neural (neurally plausible) system, and gives a finer grain to the cognitive architecture with its more sophisticated activation/latency mechanism. Johnson (1997) found two problem areas for SOAR: (1) the difficulty of SOAR to account for probabilistic move selection, and (2) the independent efforts of history and distance to goal on the likelihood of executing a move. Whereas ACT-R's control mechanism appears to be (1) well-supported by empirical data, (2) specifies the order and latency of control, and (3) is capable of dealing with complex cognitive phenomena. Irregardless of the proven multi-actor capability of SOAR, we adopt the ACT-R architecture in this research based on the cognitive criteria. Nonetheless, we will draw on the useful processes and component structure of SOAR. For instance, the graph structure of productions in SOAR—compared with ACT-R's list structure—provides better control and indexing of associations in memory.

4.4 ACT-R

The selection process has resulted in a preference for ACT-R as a model or blueprint for the implementation of a new cognitive actor architecture RBot[24], which will be discussed in the next chapter. In the remaining part of this chapter, we explain the inner workings of ACT-R in more detail.

In the last 15 years, production systems such as ACT-R (Anderson & Lebiere, 1998), 3/4CAPS (Just & Carpenter, 1992), EPIC (Meyer & Kieras, 1997), CLARION (Sun, 2003), and SOAR (Newell, 1990) have begun to truly realise the potential that Newell (1973, pp. 463–464) has ascribed to them.

The ACT-R architecture can be singled out as the one that comes closest to meeting all *levels of description* (Dennett, 1978, 1987, 1991; Gazendam & Jorna, 2002; Anderson, 1990; Jorna, 1990) of a (cognitive) actor. It can be described by three or four levels of description to predict the behaviour of the overall system, i.e. the implementation level as an approximation of the physical level (the sub-symbolic level), the functional level—the functionality of procedures interacting with declarative chunks, and the intentional level—the goals of the system or intentional module. ACT-R is a production system that specifies processes of perceiving, retrieving, storing and using information or chunks of knowledge. The three important components of ACT-R are: "[1] *rational analysis* (Anderson, 1990) [, see section 4.4.1], [2] the distinction between *procedural* and *declarative* memory (Anderson, 1976), and [3] a *modular* structure in which components

[24] In chapter 5, we implement RBot as well as a multi-actor environment, the so-called MRS (Multi-RBot System).

communicate through *buffers*" (Taatgen & Anderson, 2006, p. 7).

The declarative memory contains facts and associations between facts; the "know-what". The procedural memory contains procedures (condition-action rules) that react on patterns of knowledge stored as symbols or expressions in data-structures; the "know-how".

The modular structure creates high level independency between different types of cognitive processes. In other words, in ACT-R, independency is created with help of buffers. Communication or interaction between modules is only possible through these buffers.

Figure 4.4 presents the basic architecture of ACT-R version 5.0[25], which shows the modularity and is composed of the following: the goal/intentional module that keeps track of the goals and intentions, the declarative module for retrieving information, the manual (motor) module and the visual (perception) module. The central production system controls the interaction between the modules and reacts on patterns that are currently active in the buffers of the modules.

ACT-R's central production system contains a recognition, selection and execution cycle: recognition is implemented in ACT-R by a condition-matching function in which a goal is matched to a procedure; the selection of a potential procedure is done with help of conflict resolution[26]; and the execution step deals with the action-side of the selected production. In other words, patterns in the buffers hold representations of the external or internal world creating a context to which productions react. One of the productions is selected and fired. This is followed by an update on the buffers creating a new context for the next cycle[27].

Although the decision cycle is clearly a serial process, parallel processes are possible at the same time in different modules given the modular nature of ACT-R.

> The architecture assumes a mixture of parallel and serial processing. Within each module, there is a great deal of parallelism. For instance, the visual system is simultaneously processing the whole visual field, and the declarative system is executing a parallel search through many memories in response to a retrieval request. Also, the processes within different modules can go on in parallel and asynchronously. However, there are also two levels of serial bottlenecks[28] in the system. First, the content of any buffer is limited to a single declarative unit of knowledge, called a *chunk* in ACT-R. Thus, only a single memory can be retrieved at a time or only a single object can be encoded from the visual field. Second, only a single

[25]The version described in this chapter is ACT-R version 4.0. However, because ACT-R is undergoing many changes, some elements of version 5.0 are incorporated as well and are applicable to the design of RBot in chapter 5.

[26]As defined by Anderson and Lebiere (1998, p. 58): "On the basis of the information in the goal, ACT-R must commit to which production to attempt to fire. Because many productions may potentially fire, deciding which production to fire is referred to as conflict resolution."

[27]ACT-R is comparable to a finite state machine that contains a repeating cycle of recognise, decide and act.

[28]See Taatgen and Anderson (2006) and Pashler (1998, 1994) for a discussion on the serial bottleneck.

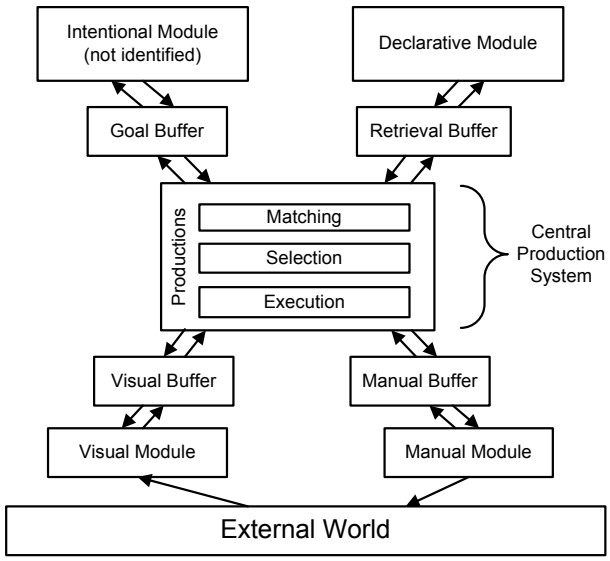


Figure 4.4: The organisation of information in ACT-R 5.0. Information in the buffers associated with modules is responded to and changed by production rules (adapted from Anderson et al., 2004).

> production is selected at each cycle to fire. (Anderson et al., 2004, p. 1038)

ACT-R has evolved and succeeded in integrating mental modules for different cognitive functions into a single theory that exists out of these modules to produce coherent cognition. The functions of the modules—perception/motor, goal buffer/module, declarative and procedural module—are elaborated in the following sections.

4.4.1 Perception and Motor module

ACT-R focuses more on higher level cognition than perception or action (Anderson et al., 2004). However, the influence of perception and action, such as perceiving new facts, or learning and acting on objects in the physical world, needs to be considered as having a significant impact on the functioning of the architecture. As previously discussed, situated and embodied cognition have shown that cognition is strongly determined by perceptual and motor processes. Where research concerns social and organisational behaviour as in this dissertation, there is a need for an actor's architecture to require properties enabling interaction and coordination with other actors.

The interaction with the environment was modelled in a specialised version of ACT-R—ACT-R P/M (Perceptual Motor) (Byrne & Anderson, 1998)—that

consists of a separate set of perception and action modules based on the EPIC[29] architecture (Kieras & Meyer, 1997). This extension enriched ACT-R with a "system for manual motor control, parallel operation of system components, and timing parameter for speech and audition" (Byrne & Anderson, 1998, p. 172). The EPIC model has proven itself to be so powerful that, nowadays, it has been adopted in all well known cognitive architectures (Taatgen & Anderson, 2006).

Although the addition of the capabilities of EPIC allows ACT-R to interact with its environment resulting in a stronger and a more comprehensive perception and motor system, the focus of ACT-R is mainly on Human-Computer Interaction and not on interaction with other ACT-R equipped actors. However, despite the lack of attention to the development of task environments for multiple actors in a Multi-ACT-R System, we have confidence that ACT-R is capable of functioning in a multi-actor environment.

4.4.2 Goal module

ACT-R's goal module has the purpose of keeping track of the intention of the actor to ensure that the goal is executed in a way that the behaviour of the actor serves the original intention of the goal, i.e. the goal module controls the flow of the task to complete the goal. Originally, in ACT-R 4.0, the goal module was equipped with a LIFO (Last In First Out) goal buffer (stack)—a traditional component in a production system—but was later abandoned, because it produced inconsistent predictions with human data. It was then replaced in ACT-R 5.0 by a "Goal State Buffer" that holds the current control state (Taatgen, 2005). There is still an ongoing debate in the ACT-R community about the ACT-R hypothesis of the single goal structure:

> Many distinct goal modules may manage different aspects of internal state... There is no reason why the different parts of the information attributed to the goal cannot be stored in different locations nor why this information might not be distributed across multiple regions. (Anderson et al., 2004, p. 1042)

In this chapter we adopt the goal stack from ACT-R version 4.0, with the optional (implementation) freedom to restrict the capacity of the buffer to a certain limit[30] and not lose compatibility with ACT-R version 5.0 that has the standard limit of one goal chunk. The goal module is composed of a LIFO buffer (stack) that is initially empty. As ACT-R runs, it pushes a main goal on the stack, which triggers procedures to either change the goal, push a sub-goal or, when the goal is solved, pop (remove) the main or sub-goal. ACT-R's goal module operates as a finite-state machine. This means that ACT-R jumps from one state to the next and when the main-goal is popped (removed), the solution is stored in declarative memory with the help of productions and finishes.

[29] Executive Process Interactive Control

[30] For example in the case of the *Minimal Control Principle*—"the model that is least susceptible to brittleness (cf. Holland, 1986) is the model with the fewest possible values in the Goal State Buffer" (Taatgen, 2005)—the goal buffer is limited to a minimal capacity, e.g. one goal chunk.

4.4.3 Declarative module

The declarative module serves as a long-term memory for declarative knowledge. The knowledge is stored as so-called *chunks* that encode small independent patterns of information as sets of slots with associated values (Anderson & Lebiere, 1998). Figure 4.5 exemplifies an addition-fact chunk. Every chunk has a type identifier (isa-slot), a *chunktype*, classifying the pattern formed in the slots of the chunk. Such a classification or mapping supports sorting and can speed up the architectural performance of locating chunks in declarative memory.

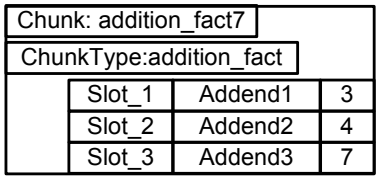


Figure 4.5: Chunk addition-fact 7.

Chunks can be created in two ways: either they are received and encoded from the perceptual module that interacts with the environment, or they are created as a result of a problem that is solved by ACT-R's deliberation process that stores popped goals (goals removed from the goal stack) as chunks in declarative memory.

The declarative memory plays a rather passive role during the resolution of a problem. It is the production memory that determines what is needed from declarative memory in order to solve a problem. The role of declarative memory is to function as a container for chunks that maintain their own activation levels. These activation levels are important for determining the retrieval time of chunks that are stored in the memory. In other words, in the process of storing chunks, activation levels tend not to impose any constraints on time needed for storing chunks. On the other hand, the retrieval of chunks in declarative memory depends strongly on the activation level of chunks; i.e. the activation level of a chunk determines the retrieval time (or the failure of retrieval) of that chunk. The processes of creating and retrieving chunks can have an impact on the activation levels of chunks and thereby on the future retrieval time of chunks in declarative memory; these processes serve as declarative learning processes and are discussed in the following section.

4.4.3.1 Activation, base-level learning and context of chunks

A strong point of ACT-R is its implementation of the functional level by ascribing sub-symbolic properties to the chunks (and procedures) in memory. The sub-symbolic properties of chunks enable the 'strengthening and weakening of information' stored in the memory.

Activation

The activation of a chunk (and the utility of procedures, explained in due

course) is one of the sub-symbolic properties that makes an actor to 'forget' and learn. The activation of a chunk is a combination or summation of (1) the base-level activation—reflecting the usefulness of chunks in the past, and (2) the associative activation—reflecting its relevance for the current context (Anderson et al., 2004).

The activation A_i of a chunk in ACT-R is defined as:

$$A_i = B_i + \sum_j W_j S_{ji} + \epsilon \qquad \text{Activation equation}^{31} \quad (4.1)$$

- B_i the base-level activation of the chunk i,
- W_j reflects the attentional weighting of the elements j,
- S_{ji} the strength of an association from source j to chunk i,
- σ the (stochastic) noise value, a noise existing out of two components: permanent noise and the noise computed at the time of retrieval based on a logistic distribution[32].

Base-level activation

Base-level activation B_i is an estimation of the log odds (of all presentations of a chunk in history) that a chunk will be used and is defined as:

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) + \beta \qquad \text{Base-level Learning Equation}^{33} \quad (4.2)$$

- t_j represents the time-difference (t_{now} - $t_{presentation}$) that the chunk was presented in memory (created or retrieved),
- n the number of times a chunk is retrieved,
- d the decay rate,
- β the initial activation upon creation of the chunk.

The equation implies that the more often a chunk is retrieved from memory (high-frequency), the more its base-level activation rises. On the other hand, the activation level of a chunk that is not retrieved at all can drop below an activation threshold level, whereby it becomes impossible to retrieve the chunk[34].

Figure 4.6 exemplifies the retrieval of a chunk at t = 10 and t = 28. It shows a small (average) increase in activation over the specified period.

As regards the base-level learning equation, Anderson and Lebiere (1998, p. 124) state that "odds of recall... should be power functions of delay, which is the common empirical result known as the Power Law of Forgetting (Rubin & Wenzel, 1996). Although it is less obvious, the Base-Level Learning Equation also predicts the Power Law of Learning (Newell & Rosenbloom, 1981)". Hence, the base-level learning equation enables the actor to learn and prefer chunks that

[31](Anderson & Lebiere, 1998, p. 70)
[32](Anderson & Lebiere, 1998, p. 73)
[33](Anderson & Lebiere, 1998, p. 124)
[34] Actually it could be possible, but only at a very large cost.

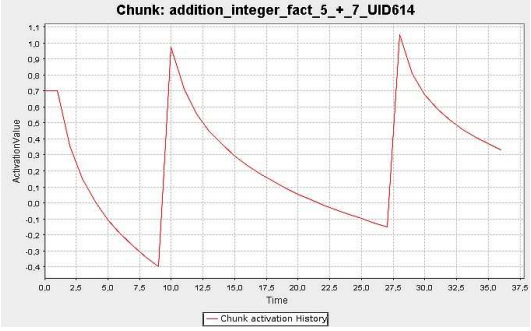


Figure 4.6: Example chunk, accessed at times 10 and 28.

are often needed for solving problems, and to neglect or forget chunks that are (almost) never needed.

The event that causes a change in the activation and presentation of a chunk takes place on three occasions (Anderson & Lebiere, 1998, p. 125)[35]:

1. The creation of a chunk:
   When a chunk is created, the first presentation of the chunk takes place and causes the initial activation. The chunk can be created in three ways:
   a. the chunk is added when the actor is pre-configured with knowledge, its birth,
   b. the chunk is perceived from the environment,
   c. the chunk is created by internal cognition, for example when a goal is solved.
2. The merging of a chunk:
   When a chunk is created and added to memory and the specific chunk already exists, the chunk is "merged" with the existing chunk, resulting in adding a presentation to the existing chunk, avoiding duplication of memory chunks.
3. The retrieval of a chunk *and* strict harvesting of a chunk:
   The production that is involved in the retrieval needs to have a strict reference to the chunk that counts as being presented, i.e. other movements of chunks, without a production intervention, do not count as a presentation.

Base-level learning has been used in many environmental experiments (Anderson & Schooler, 1991) and has been the most successfully and frequently used part of the ACT-R theory (Anderson et al., 2004).

The retrieval of a chunk from memory not only depends on base-level activation, but also on the strengths of association between chunks that (partially) match the demanded chunk.

[35]See also ACT-R tutorial Unit 4 ACT-R version 5.0: Base Level Learning and Accuracy.

Spreading activations

Several chunks in memory can be of the same chunktype in which case the contents of the chunk, the slots, become important for matching. For example, the retrieval of a chunk from memory such as an addition-fact chunk with the following slot: slotname == sum and value == 12, many answers are possible (e.g. 1+11 = 12, . . . , 11+1 = 12, but also 1+2 = 12). Hence, the activation is spread over all the (correct and incorrect) facts causing the cognitive system to have more trouble in retrieving the correct chunk from memory. At the same time, latency is increased (see the next section), because activation is reduced by the fan-effect, i.e. the effect of an activation being spread over chunks (Anderson, 1974).

Consider the retrieval of a chunk that has to be matched with the condition side of a procedure (the retrieval part) and assume a request of a chunk with addend1 == 3 and addend2 == 4; figure 4.7 visualises the activation spread.

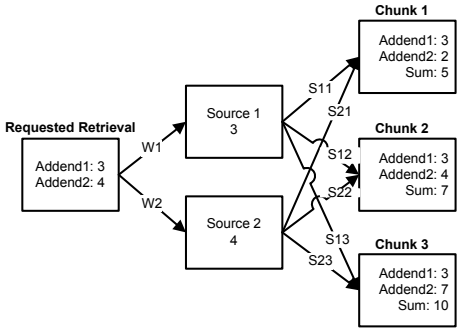


Figure 4.7: Activation spread between goal chunk and three chunks in memory.

The basic equation is:

$$SA_i = \sum_j W_j S_{ji} \qquad \text{Spreading Activation Equation (4.3)}$$

W_j the attention weight which is set to $1/n$ where n is the number of sources and causes an even distribution of strengths,
S_{ji} the strength of the association between source j and chunk i,
d the decay rate.

ACT-R has three methods of calculating the activation spread.

1. Normal matching

$$SA_{chunk1} = W_1 * S_{11} + W_2 * S_{21} = 0.5 * 1 + 0.5 * 0 = 0.5$$
$$SA_{chunk2} = W_1 * S_{12} + W_2 * S_{22} = 0.5 * 1 + 0.5 * 1 = 1$$
$$SA_{chunk3} = W_1 * S_{13} + W_2 * S_{23} = 0.5 * 1 + 0.5 * 0 = 0.5$$

2. Fan matching equation

   The general equation used for calculating spreading activation in ACT-R is the fan matching equation. Depending on the spreading of the fan-effect, every chunk gets a certain strength association based on the equation

$$S_{ji} = S - \ln(fan_j) \qquad \text{Activation Spreading Fan Equation}^{36} \quad (4.4)$$

   S_{ji} the strength of the association between source j and chunk i,
   S a parameter to be estimated (set with the maximum associative strength parameter),
   fan_j the number of chunks in which fan_j is the linked values of a slot plus one for chunk j being associated with itself.

   The S_{ji} parameter in the previous equation of normal matching is substituted with the activation spreading fan equation that results in the following values for SA.

$$SA_{chunk1} = 0.5 * (S - \ln 4) + 0.5 * 0$$
$$SA_{chunk2} = 0.5 * (S - \ln 4) + 0.5 * (S - \ln 2)$$
$$SA_{chunk3} = 0.5 * (S - \ln 4) + 0.5 * 0$$

3. Partial matching

   By not punishing chunks that are closer to the requested retrieval as much as chunks that are further away from the requested retrieval, partial matching looks not only at equality of slots in chunks, but also at the difference of slots in the retrieval, making it possible to assume that 3+2 is closer to the answer than 3+7. The equation of partial matching is:

$$SA_i = \sum W_j S_{ji} - D \qquad \text{Partial Matching Equation}^{37} \quad (4.5)$$

   The D parameter is a mismatch penalty and can be applied to compare numbers, strings and other objects. For instance, for numbers the following equation could be defined (Anderson & Lebiere, 1998, p. 78):

$$D = penalty * (|p_1 - f_1| + |p_2 - f_2|) \quad \text{Mismatch Penalty Equation (4.6)}$$

   In this case, p_1 and p_2 are the sources (source 1 and source 2) and f_1 and f_2 are the slots in the targeted chunks (chunk1, chunk2 and chunk3) and $penalty$ is the punishment parameter.
   For the case of figure 4.7, the following can be estimated:

$$SA_{chunk1} = (0.5 * 1 + 0.5 * 0) - penalty * (|3 - 3| + |4 - 2|) = 0.5 - penalty * 2$$
$$SA_{chunk2} = (0.5 * 1 + 0.5 * 1) - penalty * (|3 - 3| + |4 - 4|) = 1$$
$$SA_{chunk3} = (0.5 * 1 + 0.5 * 0) - penalty * (|3 - 3| + |4 - 7|) = 0.5 - penalty * 3$$

[36] See ACT-R tutorial Unit 5 ACT-R version 5.0: Activation and Context.
[37] (Anderson & Lebiere, 1998, pp. 76–80)

When all methods are compared, it becomes apparent that partial matching looks at the associations of chunks at slot-level and is therefore the most accurate estimator. However, the drawback is that the comparison of integers is easier than for instance the comparison of alphanumeric number three with integer 3. The partial matching method is harder to apply. Therefore, the fan equation is recommended in these cases.

Hence, in order to apply partial matching and to calculate the difference of slots of chunks, the cognitive system should provide for the estimation of that difference and to what extent that difference needs to be penalised.

4.4.3.2 Retrieval parameters of declarative chunks

ACT-R not only specifies which chunks are retrieved from memory with the help of an activation level, but is also able to calculate estimations about the probability of retrievals and the time of those retrievals based on this activation level and the threshold. For an architecture to be cognitive plausible, the conditions under which chunks cannot be retrieved anymore and the duration it takes to retrieve a chunk are important.

Retrieval Probability of a chunk

ACT-R assumes that "objects [, e.g. chunks,] are never removed from memory ..." (Taatgen, 1999, p. 43) and therefore it does not have the option to remove knowledge. Instead of removing objects, ACT-R applies a threshold value τ in the retrieval of chunks. When the activation level of a chunk falls below the threshold value τ, that chunk can no longer be retrieved. The threshold value τ has an impact on the probability of retrieving a chunk from memory. Apart from the threshold, the activation level itself and the noise parameter of the model have an effect on the probability of retrieving a chunk from memory.

This probability is given by the following equation:

$$P_i = \frac{1}{1 + e^{-(A_i - \tau)/s}} \qquad \text{Retrieval Probability Equation}^{38} \ (4.7)$$

- A activation of the chunk,
- τ threshold value,
- s stochastic variable: $s = \sqrt{3\sigma/\pi}$ where σ^2 is the combined temporary and permanent variance in the activation levels.

Probability of retrieving a matching chunk i for a specific procedure p

The matching of a chunk with a procedure depends on the base-level activation and the activation spread. Apart from that, the actual selection is influenced by the noise added to the activation. The probability of retrieving chunk i for procedure p can be approximated with the following equation.

[38] (Anderson & Lebiere, 1998, p. 74)

$$P_{ip} = \frac{e^{M_{ip}/t}}{\sum_j e^{M_{ip}/t}} \qquad \text{Chunk Choice Equation}^{39} \quad (4.8)$$

M_{ip} $M_{ip} = B_i + SA$; matching score between chunk i and procedure p, and SA is one of the matching methods; for example partial matching/fan-effect,
j total of matching chunks,
t $\sqrt{2}s$, s equals the stochastic variable in equation 4.7.

This equation expresses that when chunks with similar matching scores are in the sample and are thereby a suitable candidate for production, the probability for any of these chunks to be retrieved by the production will decrease due to competition between those chunks. Next, even though ACT-R tends to select the chunk with the highest matching score, it gives room for exploration of other chunks due to the influence of noise, i.e. ACT-R is tolerant to the occasional selection of a chunk with a lower score[40].

Retrieval time and latency

The total (matching) activation of a chunk (base-level learning, associative strength/partial matching and noise) determines the matching score M_{ip}, which individually determines the speed of retrieving a chunk from declarative memory. Chunks with higher activation have the tendency of having lower retrieval times. For instance, a question addressed to the Dutch on the name of the prime minister or president in the Netherlands gives a faster response time than a question addressed to them on the name of the president of for example Zambia. This can be explained by the fact that people are more exposed to information about their local environment than information about other countries. Hence, activation levels of chunks that point to local or familiar information are higher than those for global information.

The retrieval time (or recognition time) of a chunk is determined by the intercept time reflecting encoding and response time, the activation level and a latency scale factor. The equation is:

$$T_{ip} = I + F * e^{-A_{ip}} \qquad \text{Retrieval Time Equation}^{41} \quad (4.9)$$

I intercept time,
A_{ip} activation level or matching score M_{ip},
F latency scale factor.

[39](Anderson & Lebiere, 1998, p. 77)

[40] The equation is referred to as the Boltzmann distribution (Ackley, Hinton, & Sejnowski, 1985) or "soft-max" rule because it tends to select the maximum item but not always (Anderson & Lebiere, 1998, p. 65).

[41]This equation is adopted from ACT-R 5.0 (Anderson et al., 2004, p. 1043), a slight change compared to ACT-R 4.0 (Anderson & Lebiere, 1998, p. 80) where production strength is still included. Production strength was included as a parameter with behaviour similar to base-level learning for chunks, i.e. production execution speed was assumed to increase with practise.

The retrieval time equation shows that retrieval time behaves like a negative exponential function. It increases when the activation or matching score diminishes and decreases when the activation or matching score grows.

A procedure can have many retrieval requests to the declarative memory, and the total retrieval time is defined as the sum of all retrievals[42]. The retrieval of several chunks by a production happens one after the other, i.e. ACT-R assumes a serial processing of requests for retrieval of chunks.

The total retrieval time of a series of requests by a procedure is defined as follows:

$$T_p = \sum_i T_{ip} \qquad \text{Total Retrieval Time}^{43} \quad (4.10)$$

T_p total retrieval time of procedure p,
T_{ip} retrieval time (equation 4.9) per chunk i.

Retrieval failure

Whenever a retrieval of a chunk takes place, a retrieval failure may occur when no chunk can be retrieved from memory due to an activation level (or matching score) of the chunk below the threshold value or simply because that chunk does not exist in memory. This results in a retrieval failure of the procedure that attempted to retrieve that chunk. The time needed for the attempt to retrieve that chunk is added to the execution time of the production that causes the retrieval. The time for retrieval failure depends on the threshold value and can be calculated with the following equation:

$$T_f = F * e^{-\tau} \qquad \text{Retrieval Failure Time}^{44} \quad (4.11)$$

τ threshold value,
F latency scale factor.

In this section, we have explained the retrieval parameters of chunks that have an effect on the performance of retrieving chunks from memory. In the following section, we will elaborate on two types of declarative learning: symbolic and sub-symbolic learning.

4.4.3.3 Declarative chunks and learning: two types of learning

Philosophers have distinguished two sources of knowledge creation by (1) sensing or interacting with the outside world, and (2) cognition or the generation of knowledge by internal processes in the mind. In the same fashion, ACT-R distinguishes two types of creating a chunk: a chunk that is created by the encoding of a "perceptual object, or a chunk created internally by the goal-processing of ACT-R itself" (Taatgen, 1999, p. 43). As an exception, as previously explained,

[42] In ACT-R 5.0, this is restricted in the sense that "... only a single memory can be retrieved at a time or only a single object can be encoded from the visual field" (Anderson et al., 2004, p. 1038).
[43] (Anderson & Lebiere, 1998, p. 80)
[44] (Anderson & Lebiere, 1998, p. 87)

ACT-R does not allow duplicate chunks in memory. It increases the activation of a chunk that already exists in memory (a merger) when trying to create the same chunk.

Hence, learning of chunks can appear at two levels, the symbolic level and the sub-symbolic level:

1. Symbolic level (the creation of a chunk):
   Learning appears by the creation of new chunks: new symbolic structures are created in memory either through perception—the acquisition of new chunks, or cognition—the storing of popped goals and thereby remembering successful reasoning results.

2. Sub-symbolic learning (merging/retrieving of a chunk):

   Learning based on base-level activation B_i : the estimation of how likely a chunk is to match a production depends on the frequency and recency of its use. Thus, the more often a chunk is retrieved from memory, the more likely the chunk is re-selected relative to other chunks.

   Learning based on associative strengths S_{ji} : depending on the associative strengths and the frequency a chunk is observed, some chunks will be more strongly connected to particular goal requests than others. This type of learning approximates connectionism, in the sense that certain associations or paths from goal to chunks in memory are stronger than others (Anderson & Lebiere, 1998, p. 129).

From studying the previous equations, the notion can be made that procedures interact closely with the chunks in declarative module. All retrievals of declarative chunks are achieved by the processing of productions or the matching of the conditions of productions, i.e. the declarative module is a passive module that depends on the actions caused by the procedural module. In the next section, we explain the function of procedures by outlining the procedural module; a module that recognises patterns and coordinates actions based on these patterns.

4.4.4 Procedural module

In ACT-R, procedures occupy a separate place in memory. "The most fundamental distinction in ACT[-R] is the distinction between declarative and procedural knowledge. This distinction goes back to the original ACTE system (Anderson, 1976), and has remained through all the modifications" (Anderson, 1993, p. 18).

Procedures exist out of a condition→action (if→then) pattern. The condition responds to the goal in working memory—the current active goal in the goal stack—and specifies the chunks that have to be retrieved from declarative memory (or other modules) in order to satisfy the condition. When a condition is satisfied in a certain context, the procedure's action-side is triggered. The action-side is able to transform the goal stack by (1) removing the current active goal, (2) adding a new (sub) goal on top of the current goal, or (3) modifying

the current goal. The (possible) remaining statements on the action-side of the procedure allow for interaction with other modules (declarative module, motor module, etc.) in ACT-R, see figure 4.8.

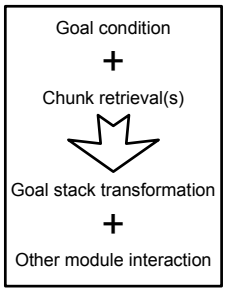


Figure 4.8: Procedure pattern.

An example of a procedure pattern is shown in figure 4.9. The procedure specifies that when the goal is to check vision, it defines a retrieval of a chunk in the perception module. And if there is any information about a car, it reacts on the perception chunk with the slots 'too close' and the driving side variable (e.g. England: =driving equals left, Europe right).

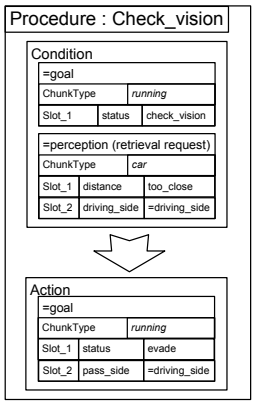


Figure 4.9: Example of a procedure pattern.

On the action side, the procedure defines a modification of the goal chunk, i.e. the slot named 'status' is set to 'evade' and the driving side variable is substituted with left or right (depending on the =driving_side value received from perception). Without these variables, information cannot be transferred from the condition-side to the action-side of the procedure. The consequence of not having variables is that for instance the check_vision procedure needs to be replaced with two procedures in which the variable =driving_side is replaced with the value 'right' and in the other procedure with the value 'left'. Hence, variables create more abstraction in procedures and empower ACT-R with the generalisation of behaviour. In case there is no matching chunk in the perception

module, the procedure is not triggered and another procedure is taken into consideration. As ACT-R has one goal stack and responds only to the goal state, the condition side always contains a single match with the current goal (=goal), while the remaining parts of the condition side can vary in the form of multiple retrieval requests and different module interactions.

ACT-R distinguishes six types of productions possible[45]:

No goal modification; only action (No Stack Change, No Goal Modification)
The action-side of the production contains only statements that allow for the execution of actions. However, in order to avoid endless looping—the production should not be selected over and over—the condition-side has to hold a test for state checking. In the case of the car example:

```
Check_vision
  =goal>
     ChunkType       running
     status          check_vision
  =perception>
     ChunkType       car
     distance        too_close
     driving_side    =driving_side
⟹
  =motor>
     ChunkType       action
     pass_side       =driving_side
```

Instead of modifying the goal (=goal), the production immediately changes a slot in a chunk in the motor module (=driving_side). The condition-side in this case tests if there still is a perception chunk with the value 'too close' in it. The production actually functions as a while loop: while there is a goal of check_vision *and* there is a car too close, undertake a motor action.

Goal Elaboration (No stack Change, Goal Modification)
This production type is the check_vision procedure in figure 4.9; it modifies the current goal, thereby creating a new (modified) goal state. The new goal state allows other productions to respond to the goal instead of the current one.

Side Effect (Push on Stack, No Goal Modification)
A goal is pushed by this type of production that triggers a procedure. For example, the check_vision procedure could be a side effect of driving the car: so now and then, the driver checks if other cars are in its vicinity.

Return Result (Push On Stack, Goal Modification)
Some procedures change the current goal, before pushing a new goal on

[45] We apply here the syntax of ACT-R (Anderson & Lebiere, 1998, pp. 41–45).

the stack. For instance, the check_vision procedure is pushed by a procedure X (a Return Result type). When the pushed goal that triggered the check_vision procedure fails, or is popped, procedure X is not selected again, because it changed the goal in the goal stack before pushing a new goal onto the stack.

Pop Unchanged (Pop Stack, No Goal Modification)
This type of procedure is applied when a goal needs to be popped and thereby returns to a higher goal in the stack. If it is the last goal in the stack, then it ends the current problem.

Pop Changed (Pop Stack, Goal Modification)
This production type is almost equal to the previous, however for storage purposes, it changes the goal that is popped, so that the goal is stored in declarative memory with extra added information.

The execution of those productions is handled by the cognitive engine, explained in section 4.4.5. It checks the goal conditions of all productions, chooses a matching production (the one with the highest utility), performs chunk retrieval(s) if needed, handles goal stack transformations, and possibly other module interactions. At the same time, the cognitive engine in ACT-R has the ability to change the values at the sub-symbolic level of chunks and procedures. First, by implementing a timer in ACT-R, the engine notifies the chunks of a time-change and enables chunks and procedures to construct a time-based history of presentations of the past (base-level learning). Second, based on events like failure or success, the cognitive engine modifies parameters at the sub-symbolic level of procedures that serve as inputs for solving a problem.

In the following section, we explain the symbolic learning that leads to new procedures in procedural memory. In the section 4.4.4.2, we describe the sub-symbolic level of procedures that registers the history of successes and failures.

4.4.4.1 Procedures and symbolic procedural learning

Prior to the start of a simulation, procedures are pre-specified to do the task for which they are designed (Anderson et al., 2004). Aside from probability matching, these procedures indicate the order that the productions ought to assume. Therefore, in the architecture that preceded ACT-R, namely ACT* (Anderson, 1983), there were efforts to discover the basic mechanisms underlying production rule learning. However, these mechanisms were not automatically adopted in ACT-R, because there was generally no evidence to support such learning mechanisms (Anderson & Lebiere, 1998). In the latest versions of ACT-R, versions 5 and 6, production learning mechanisms were sought again and adopted. Therefore, we include here the four mechanisms of ACT* (Anderson & Lebiere, 1998, p. 106):

1. Discrimination. If a rule successfully applied in one situation and did not in another situation, variants of the rule would be

generated by adding condition elements that restricted the productions to the appropriate situations.

2. Generalization. If two rules were similar, a generalized production was produced by either deleting condition elements or replacing constants by variables.
3. Composition. If there was a sequence of rules that applied in a situation, a new single rule could be formed that performed all the actions of the sequence.
4. Proceduralization. If a rule retrieved some information from declarative memory to instantiate an action to be performed, a variant of the rule could be created that eliminated the retrieval and just performed the instantiated action.

The basic motivation for these learning procedures was the observation that people became more tuned in their application of knowledge (discrimination), generalized their knowledge (generalization), came to skip steps in procedures (composition), and eliminated retrieval of declarative knowledge (proceduralization).

In the latest versions of ACT-R[46], *production compilation* (Taatgen, 1999; Anderson et al., 2004) has been introduced as a combination of "two mechanisms from ACT*, proceduralization and composition of two rules into a single mechanism" (Taatgen & Lee, 2003, p. 5). We explain procedure compilation with the help of an example from Taatgen and Lee. Consider the following three production rules to find the sum of three numbers:

Rule 1:	IF	the goal is to add three numbers
	THEN	send a retrieval request to declarative memory for the sum of the first two numbers
Rule 2:	IF	the goal is to add three numbers
	AND	the sum of the first two numbers is retrieved
	THEN	send a retrieval request to declarative memory for the sum that just been retrieved and the third number
Rule 3:	IF	the goal is to add three numbers
	AND	the sum of the first two and the third number is retrieved
	THEN	the answer is the retrieved sum

The production or procedure compilation is described as creating faster execution procedures through the incorporation of retrieved past facts as hard facts

[46]At the time of implementing RBot, production compilation was (and maybe still is) in a phase of testing, but because of its importance given in the latest versions and because production compilation refers to skill acquisition (habit of actions), we have included it as part of ACT-R and the theoretical description of the cognitive architecture.

For more discussion and application in cognitive models, we refer to Anderson (1982), Taatgen (1999, 2005) and Taatgen and Lee (2003).

in procedures (proceduralization) in combination with the merging of procedures into one procedure (composition). In the case of proceduralization, retrievals from declarative memory are not necessary anymore and the execution time of the procedure decreases. Composition is a merging process that involves creating a new procedure based on the condition side of the first procedure and adding the action side of the second procedure. For example, when a number of facts need to be retrieved, e.g. the numbers 1, 2 and 3, then there is the possibility of creating two new productions: a new rule 1—a combination of old rule 1 and old rule 2, and a new rule 2—a combination of the old rules 2 and 3. These result in the following new rules:

New Rule 1:	IF	the goal is to add 1, 2 and a third number
	THEN	send a retrieval request to declarative memory for the sum of 3 and the third number
New Rule 2:	IF	the goal is to add three numbers and the third number is 3
	AND	the sum of the first two numbers is retrieved and is equal to 3
	THEN	the answer is 6

This rule can be further specialised into a simplified single rule that combines all three rules:

Combined Rule	IF	the goal is to add 1,2 and 3
	THEN	the answer is 6

The reduction of retrievals of knowledge from the declarative memory and the reduction of rules into specialised rules occurs when similar situations occur often and the cognitive system is triggered to be more efficient in the use of resources to solve problems, i.e. reoccurring situations promote the acquisition of skills that eventually lead to habits of action. Finally, such a process will reduce problem solving time and transform a novice into an expert.

4.4.4.2 Procedures and sub-symbolic procedural learning

While symbolic learning is about creating new procedures, sub-symbolic learning is about the change of the procedure itself over time. ACT-R administers parameters for credit management of the procedures involved. In a way similar to base-level learning, the production strength equation[47] in ACT-R 4.0 is defined as the decrease of execution time of a production after frequent use. However, in versions 5.0 and 6.0, this parameter is removed. A reason for this is given by Taatgen (1999, p. 217): "...at the start of an experiment, most task-specific knowledge is still declarative. This declarative knowledge is only gradually

[47]Production strength (Anderson & Lebiere, 1998, p. 132): $S_p = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) + \beta$

compiled into production rules, providing the speed-up normally explained by strength learning".

The main difference between the selection of productions and declarative chunks is the mechanism behind the selection process. The selection (retrieval) of declarative chunks is based on activation levels, while the selection of productions is based on utility calculations.

Utility

In ACT-R, every procedure has a utility. Utility is used in order to make selection possible (conflict resolution) in cases where more than one procedure matches the current goal. ACT-R sorts procedures based on utility and selects the procedure with the highest utility. This process is called conflict resolution and will be explained in section 4.4.5 as part of the cognitive engine. The utility (or expected gain) depends on the probability of success and the associated cost of selecting a particular procedure.

The utility of a procedure is defined as:

$$U = P * G - C + \sigma \qquad \text{Utility Equation}^{48} \ (4.12)$$

- P aggregate learning probability, depending on successes and failures of the procedures (explained later),
- G goal value, the amount of effort (in time) that the actor is in willing to spend in order to achieve the goal, e.g. 10 seconds,
  C estimate of the cost (also explained later),
- σ stochastic noise variable.

The noise σ is added to the utility to create non-deterministic behaviour. Otherwise, with deterministic behaviour, the selection process of ACT-R would give no room for other productions to be executed (fired), i.e. those whose condition side matches the goal as well.

G or goal-value is the importance ACT-R attaches to solving a particular goal. There is the possibility to assign every procedure a different goal-value that influences the ACT-R engine in selecting a procedure to solve a particular goal. In most ACT-R models, the goal-value is kept equal among all procedures. An exception is the goal stack in ACT-R 4.0; the goal stack adjusts the goal-value to lower values when the goal is a subgoal of a higher goal (explained later).

The learning probability P depends on two sub-probabilities: q, the probability of the procedure working successfully, and r, the probability of achieving the goal upon the successful functioning of a procedure. P is defined as:

$$P = q * r \qquad \text{Probability Goal Equation}^{49} \ (4.13)$$

- q probability of the procedure working successfully,
- r probability of achieving the goal if the procedure works successfully.

[48](Anderson & Lebiere, 1998, p. 77); (Anderson et al., 2004, p. 1044)
[49](Anderson & Lebiere, 1998, p. 62)

Similar to probability P, the cost of a goal is divided into two parameters:

$$C = a + b \qquad \text{Cost of Goal Equation}^{50} \ (4.14)$$

a effort in time of executing the procedure,
b the estimate of the amount of time a procedure takes before the goal is achieved.

Cost C reflects the amount of time necessary for a procedure to complete its task. Cost a is the amount of time that is used to execute the procedure (retrieval cost and production execution) and all subgoals (and sub-procedures) that are triggered by the procedure until the goal that has triggered the original procedure is changed. Hence, cost a reflects the cost of search-paths in memory.

Cost b is the estimated amount of time the procedure takes to achieve the completion of the goal and to solve a (sub) problem. Such costs can create problems, because particularly in tasks that take a long time, e.g. two weeks, the effort invested into accomplishing the goal can be large. Upon the resolution of a problem, the utility of a procedure can drop so dramatically that its re-selection is no longer justified. Future investigations of this problem can provide solutions for time-consuming tasks by creating a dependency between the goal-value (or cost) and the type of task, i.e. the goal-value could be based on the motivation for and expectancy (and passed experience) of a certain task to which it is related.

The q, r, a and b parameters are affected by experiences over time and allow the cognitive system to operate at all times to optimise the adaptation of the behaviour of the organism (Anderson, 1990, p. 28). We initially explain how successes and failures affect learning. We then discuss the influence of efforts on learning.

Probability q and r

Probability q and r describe two types of success-ratio of production usage. They reflect the history of successes and failures of a production. q is the success-ratio of executing the procedure, including all retrievals and the execution of, if present, any subgoals. r is the success-ratio calculated over the achievement of completing a goal, after all of the subgoals are solved and the current task and its related goal are met and popped (explained in 4.4.5). q and r are defined as:

$$q, r = \frac{Successes}{Successes + Failures} = \frac{S}{S + F} \qquad \text{Probability Learning Equation}^{51} \ (4.15)$$

S experienced successes,
F experienced failures.

In a simulation setup, the modeller sometimes brings in knowledge in the form of prior past successes and failures. These are combined with current successes and failures, experienced upon the start of a simulation. To make this

[50](Anderson & Lebiere, 1998, p. 62)
[51](Anderson & Lebiere, 1998, p. 135)

distinction clear, the equation is specified as follows:

$$q, r = \frac{\alpha + m}{\alpha + \beta + m + n} \qquad \text{Probability Learning Equation}^{52} \quad (4.16)$$

- α prior successes (default 1),
- β prior failures (default 0),
- m experienced successes,
- n experienced failures.

Cost a**,** b

As with the success-ratio, costs are also divided into two types. Cost a reflects the cost that is expected to occur when executing a procedure, which includes the rule's retrieval time(s), plus the procedural execution time and additional procedure action time where the action time includes a fulfilling of a subgoal (elaborated in 4.4.5).

Cost b starts the moment cost a stops. Cost b reflects the cost remaining after the procedure is executed until the goal is achieved successfully or is dropped owing to a failure (see also figure 4.11). Cost a and b are defined as:

$$a, b = \frac{Efforts}{Successes + Failures} = \frac{E}{S + F} \qquad \text{Cost Learning Equation}^{53} \quad (4.17)$$

- E experienced efforts,
- S experienced successes,
- F experienced failures.

The differences, compared to the Probability Learning Equation (4.15), are that successes are replaced with the effort of executing and fulfilling a task and those efforts are a summation of all the efforts experienced by a procedure, no matter if the experience is a failure or a success. Similar to the Probability Learning Equation (4.16), we can specify the prior successes and failures, a factor z that represents prior influence of efforts, the summation of efforts experienced during the simulation, and the experienced successes and failures of a procedure during the simulation. This can be expressed with the following equation:

[52](Anderson & Lebiere, 1998, p. 135)
[53](Anderson & Lebiere, 1998, p. 135)

$$a, b = \frac{z + \sum_{i}^{m+n} effort_i}{\alpha + \beta + m + n} \qquad \text{Cost Learning Equation 2}^{54} \quad (4.18)$$

z	total prior effort (for a default 0.05, for b default 1),
$\sum effort_i$	summation of all experienced efforts (effort typically measured in time),
α	prior successes (default 1),
β	prior failures (default 0),
m	experienced successes,
n	experienced failures.

The similarities between the success-ratio crediting and the cost crediting is that success-ratio q and cost a are credited at the same time. The success-ratio r and cost b are also credited at the same time. The details about the credit assignment are explained in section 4.4.5. In addition to the calculation of successes, failures and efforts, ACT-R applies discounting: events and efforts that happened in the past have less of an impact than present events and efforts.

Event & Effort Discounting

In the previous equations for calculating the success-ratio and the cost of efforts, no time component was included. Under such a condition, events and efforts in the past are equally weighted as the ones experienced at present time. However, people are often more aware of the impact of present experiences (of the same procedure) than experiences encountered in the past. ACT-R has a mechanism that takes care of discounting past experiences by implementing an exponentially decaying function that is similar to the base-level learning equation. The equation for discounting successes and failures is:

$$Successes, Failures = \sum_{j=1}^{m,n} t_j^{-d} \qquad \text{Success Discounting Equation}^{55} \quad (4.19)$$

m	number of successes,
n	number of failures,
t_j	time difference, now - occurrence_time of the success or failure,
d	decay rate (default: 0.5).

This equation can be substituted in equations (4.15), (4.16), (4.17) and (4.18). This substitution results in a set of new functions for calculating learning probability and cost that depend on the decaying of successes and failures over time.

The effect of the equation on the selection mechanism is that there are more exploration possibilities caused by the decay of past successes and failures. For example, it is possible to give different decay rates for successes and failures, i.e. when the decay rate of failures is lower than that of successes, ACT-R tends to forget negative experiences more slowly than positive experiences.

[54](Anderson & Lebiere, 1998, p. 136)

[55](Anderson & Lebiere, 1998, p. 141) & (Anderson & Lebiere, 1998, p. 265)

The same kind of discounting can be applied to efforts discounting:

$$Efforts_{a,b} = \sum_{j=1}^{m,n} t_j^{-d} Effort_j \qquad \text{Effort Discounting Equation}^{56} \ (4.20)$$

a	cost a,
b	cost b,
m	iterator representing efforts belonging to cost a,
n	iterator representing efforts belonging to cost b,
$Effort_j$	Effort at iteration j (in time units),
t_j	time difference, now - occurrence_time of the success or failure,
d	decay rate (default: 0.5).

The equation expresses that efforts associated with cost a and b are discounted over time resulting in past efforts to be forgotten over time. The efforts in equations (4.17) and (4.18) can be substituted with the equation stated in equation (4.20). This creates new equations that include the decay of past efforts.

Discounting creates the possibility that procedures—that are not used for a long time, because they are credited with high cost—still can be selected in the future. Such an opportunity is possible when other procedures create errors in an environment that demands other capabilities of the actor. When these capabilities are in the procedures that are outranked owing to their high cost, they have a higher chance of being selected in the new situation when efforts have decreased over time. Otherwise, these procedures may not be selected again, because the punishment of a high effort never decreases over time.

The following section explains the cognitive engine that controls the matching of procedures, the selection by means of *conflict resolution* and the execution and credit assignment at the sub-symbolic level.

4.4.5 The cognitive engine

The cognitive engine defines a clever mechanism by connecting the procedural module to patterns in other modules of ACT-R. The mechanism consists of cycles and each cycle consists of a certain number of steps. In other words, the engine of ACT-R is constructed as a finite state machine defining a loop, which has the following flow:

1. The condition parts of all procedures are matched to the pattern of the current goal. This results in a list of potential candidates from which one is selected for execution in the end. However, if no candidates are found, the current goal is removed and the next goal in the stack becomes the current goal. The engine returns to step 1 until there are no goals left anymore, in which case, the engine halts.

[56](Anderson & Lebiere, 1998, p. 141)

2. The conflict resolution occurs when more than one procedure matches the goal. In that case, the procedure with the highest utility is selected.

3. The selected procedure is executed (fired). However, when there are any remaining conditions left—such as retrievals from other modules (e.g. declarative memory or perception)—then these are processed first. When these retrievals result in retrieval errors, the production fails prior to processing the action side. The engine then goes back to step 2.

   If the engine succeeds in processing the condition-side without any failures, then the action-side of the procedure will be handled. The action-side contains actions that can change the contents of the goal stack and/or interact with other modules.

4. When execution succeeds, or failure takes place, the engine starts the accounting of values to the relevant sub-symbolic parameters of the procedures and chunks.

5. There is a return to state 1 upon the existence of a goal in focus (current goal), otherwise the engine exits. (halt)

One side condition is necessary for ACT-R to avoid endless looping of the engine: the goal module must always change[57] to avoid loops in a single cycle. However, endless cycles can occur when the next procedure puts the goal module in its previous state. If there is no escape, ACT-R will go into an endless loop[58].

The ACT-R engine and the interaction with the memory modules is shown in figure 4.10.

The ACT-R engine reacts based on the contents of the current goal. The current goal is matched with all the procedures. The procedures that match the current goal are candidates for firing. When there is more than one procedure that is a candidate for execution, conflict resolution takes place. Conflict resolution is a process that compares the utilities of all candidates and selects the procedure with the highest utility.

After a procedure has been selected, the necessary retrieval requests to memory are made. When no retrieval failures are reported, the action side of the procedure is executed. Change such as popping, modifying or pushing a goal in the goal stack occurs on the action side of the procedure.

As mentioned before, ACT-R states that popping (removal) of a goal occurs with placing the popped goal from the goal stack into declarative memory. In this way, new chunks are learnt based on cognitive activity. Besides the creation of declarative chunks, new productions are created with the help of production compilation, which is an autonomous process inside the engine that creates new procedures out of old procedures and/or chunks in memory.

The inner working of the ACT-R engine, i.e. the five steps of the ACT-R engine, will be explained in the following sections.

[57] There is one exception (see section 4.4.4): the procedure type *no stack change, no goal modification*.

[58] This strongly depends on the developer who is constructing the procedures. There is not yet an escape mechanism that prevents an ACT-R to avoid entering a vicious loop.

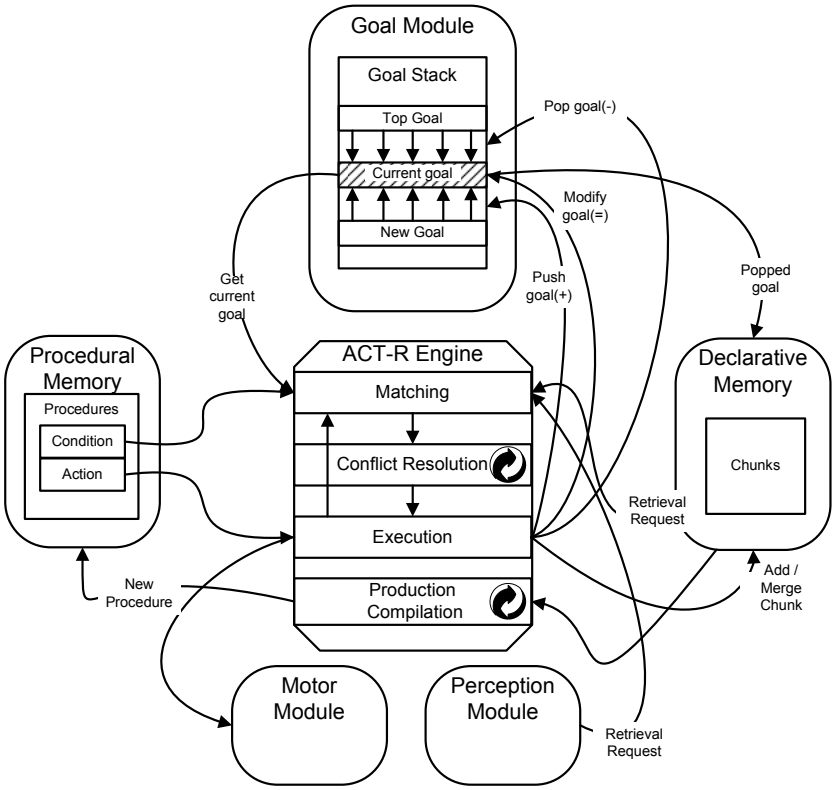


Figure 4.10: ACT-R engine and interaction.

4.4.5.1 Matching (Step 1)

The ACT-R engine compares the goal chunk with the =goal statement in the procedure condition. It checks all the procedures and sees if a match is there. When a match is found, it is added to a list of procedures that have the potential to fire. In ACT-R, the matching process does not involve a check for the success or failure of retrieval requests. This is handled in the execution step.

4.4.5.2 Conflict Resolution (Step 2)

After matching all procedures, a set of matching procedures is obtained. ACT-R, in contrast to SOAR, only fires one procedure at a time. The matching procedures are seen as being in conflict with each other and a process, *conflict resolu-*

tion, has to make clear which procedure will eventually fire. Figure 4.11[59] shows the selection of a procedure (P4) based on conflict resolution.

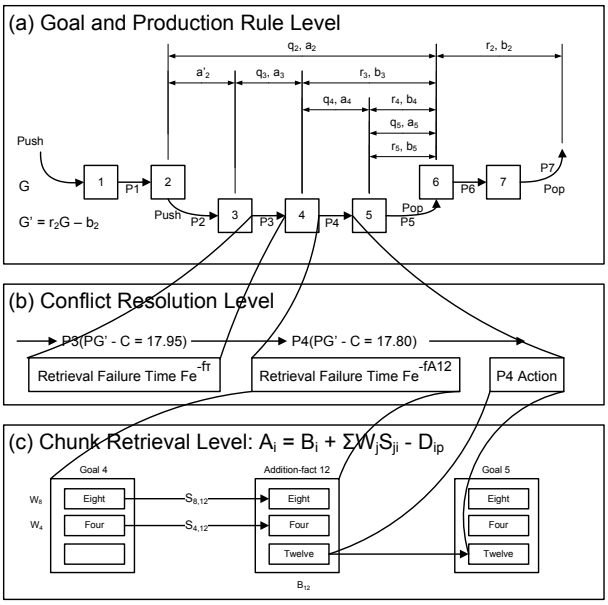


Figure 4.11: Conflict resolution / credit assignment (Anderson & Lebiere, 1998, p. 59).

As regards conflict resolution, the procedures that are important in figure 4.11 are procedures P3 and P4. Procedures P3 and P4 are in the potential procedure list, of which procedure P3 is the winner, because its score of 17,95 is higher than 17,8. However, after trying the retrieval request on the conditional side of procedure P3, the procedure fails and procedure P4 is the next in rank to be selected. This procedure has no retrieval errors and can be executed (step 3) without any problems[60].

[59]The process of solving a goal is as follows: the goal G is pushed on an empty goal stack; this leads to state [1]. Production P1 is triggered by this state and changes state [1] into state [2]. Production P2 pushes a new goal on top of the goal stack, and this leads to state [3]. The goal-value [*G*] is reduced to [*G'*] when a goal is pushed; 'deeper' goals are less likely to bring success in solving the problem.
State [3] triggers P3, which tries to retrieve a fact from declarative memory. However, as can be seen at (b), the conflict resolution level, P3 results in a retrieval error. In that case, the next procedure, P4 from among possible candidates, is selected for firing (thus, state [3] equals state [4]). At the chunk retrieval level (c), P4 succeeds in finding a chunk (addition-fact12) and changes state [4] in state [5] by filling in an empty slot with 'Twelve'. Production P5 pops the goal and brings state [5] to state [6]; back to goal level G. Production P6 changes state [6] to state [7] followed by production P7 that pops the goal and empties the goal stack.

[60]In case no procedure can be found at all, the goal (e.g. goal4) will be dropped by ACT-R with a failure.

4.4.5.3 Execution (Step 3)

In step 3, if any statements remain on the condition part of the procedure, they are checked out by trying to retrieve those statements from declarative memory or other defined modules. This means that there is a set of retrieval requests for chunks in memory that has to be executed. If that step fails (procedure P3), the ACT-R engine goes back one step and selects, in this case, procedure P4 for execution. The condition side of procedure P4 appears to have no problems, so the action side of P4 is executed and state [4] is transformed to state [5] with the third slot of the goal (Goal 5) filled with number 'Twelve'.

4.4.5.4 Credit assignment (Step 4)

Credit assignment is the process of assigning values at the sub-symbolic level. Examples of credit assignments are (1) values of goal levels (G and G'), (2) the sub-symbolic values q and a: the probability and time for executing a procedure, (3) the procedure values r and b—after execution, the probability of achieving the goal and the estimate of the amount of time a procedure takes before the goal is popped, and (4) notifying chunks when they are retrieved for the purpose of base-level learning.

Goal level credit assignment

ACT-R makes a distinction in goal values at different levels of goal stack execution. The stack in figure 4.11 appears to be two levels deep. ACT-R calculates the goal-value at level 2 (G') by discounting the goal-value at level 1. The goal-value is calculated based on the expected future effort and success of achieving the subgoal. The equation for estimating the goal-value of a goal at a lower level is.

$$G' = G_{level_{x+1}} = r * G_{level_x} - b \qquad \text{Goal Discounting}^{61} \quad (4.21)$$

G_{level_x}	goal at level/depth x of the stack,
r	probability of achieving the goal if the procedure works successfully,
b	the estimate of the amount of time a procedure takes before the goal is achieved.

> G' is *discounted* from G to reflect uncertainty (r) and cost (b) in achieving the goal even after the subgoal is achieved. Thus, G' reflects the maximum amount that should be expended in achieving the subgoal and this is the value attributed to this subgoal. Thus, the value assigned to a subgoal depends on the context in which it occurs. (Anderson & Lebiere, 1998, p. 63)

ACT-R shows that the lower a goal level is, the higher the probability that the problem is unsolved at that level. We argue that this is cognitive plausible, because human beings are boundedly rational and are (normally) not able to

[61] (Anderson & Lebiere, 1998, p. 63)

memorise goal structures with many subgoals. We state that the goal-value can reflect the capability and motivation for solving goals.

Credit assignment of q and a
There are two possibilities following the execution of the procedure:

1. No subgoal is pushed and the ACT-R engine immediately can assign credits to q and a (figure 4.11, q_3 and a_3). When a procedure (P3) causes retrieval errors and fails, the number of failures (F) for q and cost a is increased with 1 and the associated retrieval failure time is estimated as $Fe^{-f\tau}$ (equation (4.11)). However, when the retrieval requests are successful (P4), both q and cost a are credited success and the retrieval time becomes Fe^{-fA}.

2. If a subgoal is pushed by a procedure (figure 4.11, P2), credit assignment (a failure or success) for this procedure takes place the moment this subgoal is popped (P5 pops the goal). Cost a of P2 is based on the successes or failures and the summation (total time) of all $efforts(a)$ of the procedures that were involved in completing the subgoal.
$effort(a_2) = effort(a'_2) + effort(a_3) + effort(a_4) + effort(a_5))$.

Credit assignment of r and b
Credit assignment to r and b takes place in the following cases:

1. The procedure pops the current goal (figure 4.11, P5 & P7). When the procedure pops the current goal, ACT-R assigns r and b a success or effort similar to that assigned to q and a of that procedure, e.g.
$< (S)r_5, (S, effort)b_5 > = < (S)q_5, (S, effort)a_5 >$

2. The procedure is followed by another procedure at the same level. For instance, procedure P3 is followed by P4 at the same level. In this case r_3 and b_3 cannot be assigned immediately to P3, only when the goal at the level of P3 is popped. In figure 4.11, procedure P5 pops the goal at the level of P3 and at that point, credits to r and b to all procedures (P3, P4 and P5) at that level can be assigned.

   In the case of P3, there are two possibilities: The successes of r_3 and b_3 depend on P5; if P5 succeeds in dropping the goal, then P3 will be credited with successes: $(S)r_3 = (S)b_3 = (S)q_5 = (S)r_5 = (S)a_5 = (S)b_5$

   A special case occurs when P5 fails and no other procedure matches, then the goal fails and q_5, r_5, a_5, b_5, r_3 and b_3 are assigned a failure:
   $(F)r_3 = (F)b_3 = (F)q_5 = (F)r_5 = (F)a_5 = (F)b_5$

   In both failure or success cases, the effort of b_3 depends on the effort of procedure P4 (cost a_4) and the effort of procedure P5 (cost a_5)
   See figure 4.11: $(effort)b_3 = (effort)a_4 + (effort)a_5$

Credit assignment of procedure execution
The only remaining effort that needs to be added to every procedure is the effort taken by the cognitive engine to execute the procedure, i.e. during every execution of a procedure, a standard execution effort of 50ms is added to the efforts of the fired procedure ($effort(a'_2)$, under the assumption that procedure P2 has no retrieval of chunks).

Assigning of presentations of retrieval or merger to chunks (base-level learning)
In the execution phase, the retrievals have to be registered at the sub-symbolic level of chunks as well as the merging of chunks when goals are popped or when new chunks from perception are encoded and merged.

4.4.5.5 Check end-state (Step 5)

In Step 5, which can also be seen as Step 0, the ACT-R engine inspects the goal stack to see if there are any goals left to fulfil: if a goal is present, the engine uses that goal for the matching of procedures and moves to step 1. If there are no more goals, the engine stops. (on halt)

The description of the core components given thus far is sufficient to start the implementation of simulation software, i.e. see RBot in the next chapter, which is dedicated to the study of organisational behaviour of a collection of cognitive plausible actors.

Prior to the design and implementation of RBot, we evaluate ACT-R and explore what is missing and what adaptations are necessary for it to fulfil the requirements for the construction of a Multi-Agent System that is able to simulate social and organisational behaviour.

4.5 Evaluation of ACT-R

The evaluation of ACT-R will be based on the three research questions mentioned in chapter 1. The research questions address what is necessary for the implementation of a cognitive agent-based social simulation tool. In order to develop such a tool with the help of ACT-R, we first evaluate ACT-R and suggest directions, changes or extensions that are needed for ACT-R to function as a cognitive plausible (4.5.3) and socially situated (4.5.2) Multi-Agent System (4.5.1).

4.5.1 Multi-Agent System

The first question (1.1) addressed what type of model was suitable for explaining interactive social behaviour. We have argued that a Multi-Agent System (MAS) suffices as a model and a methodology to explain interactive social behaviour. Although ACT-R is not a Multi-Agent System and is primarily focused on the individual and its interaction with the environment (and not on interaction with other actors), there have been some developments of ACT-R agents interacting in small teams within a spatial domain and real-time environments (Best

& Lebiere, 2003, 2006). These developments have "demonstrated that the same framework for action and perception at the cognitive level used in the ACT-R agents [...] can be used to control behavior in both virtual and robotic domains" (Best & Lebiere, 2006, p. 216). This shows that ACT-R is capable of interacting and communicating in a virtual as well as a physical environment and can serve as a plug-in for other environments.

A sustainable development of a Multi-ACT-R model is however still lacking owing most likely to the cognitive psychological roots of the ACT-R community. On the other hand, Multi-Agent Systems are part of Distributed Artificial Intelligence, which is a community that is interested in problems solved by coordinating interactions between different actors distributed over several threads, on the same computer or distributed over several computers in a network. In order to bridge the gap between MAS and ACT-R, some modifications are necessary to pose ACT-R as a serious candidate for Multi-Agent Systems.

Adjusting the design of ACT-R: integration in a MAS.

The first step for ACT-R is (1) the need to have an artificial (simulated) task environment in order to construct various multi-actor environments and interaction scenarios. These interactions can be accomplished by the exchange of signals and signs between the actor, the (physical) environment and other actors, as described by the model of Best and Lebiere (2006).

In the next chapter, we introduce RBot—a cognitive social daughter of ACT-R—and MRS (Multi-RBot System) that implements an artificial task environment. The creation of such an artificial task environment that is suitable for multi-agent interaction suggests a need for a set of ACT-R agents that can be distributed over a network, i.e. a network that consists of heterogeneous computer systems[62]. This requires (2) an ACT-R agent that is portable to different computer platforms, and (3) requires an Agent Communication Language (ACL) that enables ACT-R agents to communicate with each other and if possible with other types of actors.

These are three adjustments we have applied to ACT-R, which have resulted in a new cognitive actor RBot. The first and radical adjustment that fulfils requirement (2) is the choice for implementing ACT-R in the programming language JAVA™. JAVA is an object oriented programming language that allows the execution of programs on a virtual machine that can be integrated in many well known flavours of operating systems[63]. The most important reason for the shift to JAVA is most likely its advancement in conjunction with the Internet, the large user community and the immense library of tools that can be integrated into the development process. The second adjustment we applied to ACT-R and which satisfies requirement (3), is the implementation of message-based communication with the help of speech acts or an Agent Communication

[62]This is a rather strong implication; we want to address the heterogeneity of systems that are connected to the Internet. Although, theoretically, a MAS could run on a single machine, we assume that future applications of ACT-R and RBot will run on distributed computers.

[63]ACT-R is programmed in LISP and also runs on different platforms. However, LISP's popularity is not as widespread as JAVA; in the sense that it is well recognised in the AI community, but not beyond. Therefore, JAVA is a good candidate for introducing ACT-R *and* RBot to other communities.

Language. An ACL such as FIPA (FIPA, 2002) allows actors to exchange information according to a format or code that is known to all agents involved and thereby supports 'open' communication between agents of any kind. The third adjustment is the implementation of a generic artificial task environment; MRS (Multi-RBot System). The MRS is a simple server system that represents a centralised artificial task environment and facilitates physical and communicative interaction between agents, i.e. the server has the purpose of providing an artificial 'physical' task *and* a communication environment that allows actors to perceive objects and other actors, and of supporting the exchange of signals and signs. The next step is the step of social interaction and distributed problem solving with the help of representations of (joint) tasks, social structures and roles in the minds of actors.

4.5.2 Social behaviour: social interaction and social constructs

The second research question addresses the requirements that are necessary for an actor to exhibit (stable) social behaviour and that such behaviour requires the creation of shared knowledge, social construction and semiotic resources that include social structures, institutions and habits of action.

We refer back to the levels of description where we distinguish levels varying from the physical or biological to that of social.

In a similar vein, Newell (1990) describes a time scale whereby different bands are identified for human action as shown in table 4.3.

Time scale of human action			
Scale (sec)	Time Units	System	World (theory)
10^7	months		
10^6	weeks		**SOCIAL BAND**
10^5	days		
10^4	hours	Task	
10^3	10 min	Task	**RATIONAL BAND**
10^2	minutes	Task	
10^1	10 sec	Unit Task	
10^0	1 sec	Operations	**COGNITIVE BAND**
10^{-1}	100 ms	Deliberat Act	
10^{-2}	10 ms	Neural circuit	
10^{-3}	1 ms	Neurons	**BIOLOGICAL BAND**
10^{-4}	100 μs	Organelle	

Table 4.3: Time scale of human action (Newell, 1990, p. 122).

> Starting from the bottom, there is the *biological band* of three levels—neurons; organelles, which are a factor of ten down from neurons; and *neural circuits*, a factor of ten up...Above the biological band, there is the *cognitive band.* Here the levels are unfamiliar—I've called them the *deliberative act, cognitive operation, and unit task*...

> Above the cognitive band lies the *rational band*, which is of the order of minutes to hours. All of these levels of the rational band are labelled the same, namely, as *task*. Finally, even higher up, there lies something that might be called the *social band*,... (Newell, 1990, pp. 122–123)

ACT-R, similar to SOAR, focuses on the individual and more specifically at the cognitive band and the rational band[64]. At the cognitive band, productions fire on average at every 100 ms (Anderson & Lebiere, 1998, p. 13) and the rational analysis takes place at the rational band (10^2-10^4). According to the time scale of human action social behaviour lies at 10^5-10^7 sec (2,8 hr - 116 days). There is a gap between the social band, the rational band and the cognitive band, varying from 10^2 (rational band) till 10^6 (cognitive band, i.e. production level) that needs to be bridged, i.e. social behaviour needs to be explained by behaviour of the individual (the cognitive and rational band) and the other way around.

Adjustment of ACT-R to allow for social interaction

In order to bridge this gap, we have argued that a semiotic level—see chapter 1 & 3—is necessary between the individual and the social level. We do this by defining the *social construct*[65] as a (social) representation in the mind of the actor, or as documents or artefacts in society, reinforced by their frequent use resulting in habits of action or social norms.

The implementation of a multi-agent task environment allows one to simulate and study social behaviour and interactions between individuals. ACT-R has the capacity for storing chunks or rules that depend on the social context or interactions with others. However, ACT-R is not designed for the purpose of explaining social behaviour. In other words, there are no social properties or mechanisms defined. Therefore, we have created the social construct as a separate unit of (social) knowledge. The social construct enables one to define social situations in which the actor has to obey or is permitted to follow certain rules that are present in a society or culture. The implementation is carried out by creating a special module in which social constructs are entities that influence certain aspects of behaviour. Imagine, for example, that a certain situation does not allow for certain goals or the execution of certain rules.

The mechanism that is adopted for implementing social constructs is the subsumption architecture (Brooks, 1986); a multi-level system in which the upper (normative) level is a social construct that influences behaviour at the lower level(s). The concept of social construct allows us to model social situations, emergent social behaviour and interactions with other actors, which is impossible to accomplish in ACT-R.

[64]The biological band is not the main focus of SOAR, ACT-R *and* this dissertation; for a description, see (Newell, 1990, p. 123).

[65]See chapter 3 for an explanation about the social construct.

4.5.3 The rational and functional level

The third research question addresses the kind of actor that can plausibly handle signs, relations and social constructs. We argue that a cognitive architecture like ACT-R is a physical symbol system that is able to handle signs and, as explained in the previous section, can handle and store social constructs.

Adjustments at the rational level: goal deliberation

ACT-R is an architecture that uses a rational approach (Anderson, 1990, p. 30) to study human behaviour. It does this by specifying what the goals are of the cognitive system, developing a formal model of the environment to which the system is adapted and making minimal assumptions about computational limitations such as the assumption of short-term memory (Anderson, 1990, p. 29).

The structures above the atomic components (declarative chunks and productions) are goal structures that provide a "chemical structure" (Anderson & Lebiere, 1998, p. 13). In ACT-R 4.0, there was a goal stack that provided such a goal structure, but this was abandoned in ACT-R 5.0, because it did not conform to predictions found in experimental data. We argue that, because there is not an intentional module defined in ACT-R, it is unclear how one can incorporate the "chemical structure" of goals in a plausible way. Although we do not provide experiments that can give directions for cognitive plausible structures, we assume that structures such as tree-like structures and a deliberation module similar to a Beliefs Desires Intentions[66] module or a motivational subsystem (Sun, 2003) is necessary for the maintenance of (social) goals.

Interaction with other actors requires more control over goals, because many collaborations with others actors depend on planning, commitment and exchange of goals. In RBot, discussed in the next chapter, we have implemented a goal list (a simple FIFO buffer) that provides the opportunity to exercise more control over the processing of goals and is able to address issues of planning or dual tasks, that is:

> The goal list can handle more complex or more subtle situations that the goal stack cannot easily handle. For example, in order to schedule several activities at the same time, such as carrying on a conversation while searching for a file in the desk drawer, the priorities of different goals, such as "continuing conversation" and "continuing file search", can change dynamically. As the delay time since a question was asked grows, the goal of "continuing conversation" becomes increasingly more important. On the other hand, when the conversation drags on, the goal of "continuing file search" may become dominant again. This kind of alternation of goals may be difficult for the goal stack to handle. (Sun, 2003, p. 50)

[66] The BDI (practical reasoning) theory is not applied in this dissertation, because it operates mainly at the rational level. However, the BDI theory can give alternatives or suggestions for the implementation of an intentional module (cf. Bratman, 1987, 1990; Rao & Georgeff, 1995; Wooldridge, 2000).

Sun (2003)'s architecture CLARION focuses more on the individual system than on cooperation with other actors. He elaborates on the problems that occur when there is not a separate structure that handles goals. He promotes the implementation of a motivational and meta-cognitive system that deals with the priorities of goals. We encountered similar problems when we assigned multiple goals to the actor. Therefore, we also incorporated a simple goal structure or list in RBot.

Adjustments at the cognitive level

The adjustments at the cognitive level are minor adjustments, compared to the adjustments at the higher level. As the next chapter shows, many mechanisms and components defined in ACT-R are not changed. In this manner, we probably maintain the cognitive plausibility of ACT-R that is seen as a necessary outcome of thorough testing over the last 20-30 years. Nonetheless, we will make a few slight adjustments as regards the memory structure, the structure of productions, and issues related to the base-level decay and estimation of efforts.

Memorymap and memory structure ACT-R creates a clear distinction in several memory modules that each have a certain functionality. In RBot, we have created the memorymap as a reference container for chunks. Some of the chunks in the memorymap are assigned to be an entrance to the contents of the memorymap. The other chunks in the memorymap are accessible via the entrance chunk, i.e. the other chunks have links with the entrance chunk and can only be reached by the entrance chunk. Because of the fact that a memorymap maintains a collection of references, chunks can be member of several memorymaps at the same time. Secondly, there exist the possibility of adding links or paths (of association) between chunks that are in different memorymaps.

In the next chapter, we will explain more thoroughly about this adjustment of memory management. The adjustment gives more flexibility in creating memory structures compared to ACT-R.

Structure of productions ACT-R distinguishes a production and a declarative memory that contain units of knowledge: productions and declarative chunks as containers of knowledge. RBot distinguishes a procedural and declarative memory as well. However, an attempt is made to reduce the units of knowledge to a single unit: the chunk. When we carefully study the syntax of a production in ACT-R (Anderson & Lebiere, 1998, pp. 31–34), we can recognise patterns of (goal) chunks in the production itself. Such a pattern (ibid., p. 34) can be a pattern-chunk (a chunk that contains variables) and these pattern-chunks can be linked together to form a production.

In other words, the production can contain a condition- and an action-chunk that is linked with pattern-chunks—a network of linked chunks—that together form the production. The production is a chunk consisting of chunks (see the next chapter for a detailed explanation).

Base-level decay The decay parameter is a problem in the sense that the parameter has to be adjusted depending on the length of the experiment (see Taatgen, 1999).

> A fast decay of 0.5, which is the official recommended value of the parameter, turns out to work best for decay within an experimental session. A slow decay of 0.3 is necessary for experiments in which an hour, a day or a week passes between experimental sessions, else ACT-R will forget all it has learnt. (ibid., p. 217)

Especially in experiments on social interaction that can take longer than a (simulated) week, the decay rate is too high. Alternative solutions should be provided to prevent the loss of social constructs that are formed in interactions occurring in the social band, a time band that can surpass hundred days.

A possible solution can be the adjustment of the decay function. For example, after an interaction or an experimental session has ended, a lower decay parameter becomes active that slows down the decaying process; see Taatgen (1999, p. 217) and Anderson, Fincham, and Douglass (1999, p. 1133). In our experiments, we assume that social constructs are habits of action and are somehow similar to productions, i.e. in our current version of RBot social constructs are not subjected to decay. Therefore we are not confronted with this problem in our experiments.

Estimation of effort The cost or effort of reaching a goal also needs to be considered. In ACT-R, the maximum effort that a task takes is between 10 and 15 seconds. However, when a task (and goal) lasts longer, and especially in social interaction and experiments where the task can take a long time before it is solved, the impact of the effort on the cost of the utility to reach a goal is not realistic. For example, when an actor travels from A to B in about 15 minutes, the cost of reaching that goal is high compared to tasks that take a shorter time. The next time the actor tries to solve the goal, it will not select the procedure that responds to the goal. This is not due to the procedure being unsuccessful, but simply the fact that it will cost too much effort to even think about executing the procedure.

The utility function of the procedure is composed of (1) the probability that a goal will succeed and (2) the cost of a successful goal. The problem of cost b we encountered in our models (and especially huge efforts) points to a clear dramatic drop in the utility of the procedure. We solved this problem roughly by limiting cost b to a maximum of 15 seconds. However, other solutions are possible. For instance, another solution could be to estimate the level of cost for achieving the goal in the beginning (a maximum aspiration level) and to compare the realised effort with the aspiration level, which results in a cost function that reflects relative cost. A combination of this solution with a motivational subsystem, which influences the goal-value, could be a good candidate. For instance, if there is a lot of repetition of efforts and the amount of effort invested results in an actor's fatigue, then the motivation may drop. This can have a negative impact on the goal-value of the procedure, encouraging the actor to take a break.

4.6 Discussion

In this chapter, we focused on the mind of the individual, i.e. on cognitive science, cognitive architecture, and specifically ACT-R. We adopted ACT-R as a cognitive architecture because it has proven its cognitive plausibility over the last 20-30 years. The analysis of the individual based on levels of description has introduced the notion that an intelligent actor should be a physical symbol system; a system that is a necessary and sufficient condition to exhibit general intelligent action (Newell, 1980). In other words, an actor should possess a cognitive architecture containing representations, and cognitive and functional mechanisms.

Apart from the notion of physical symbol systems or the classical approach, we explained the importance of two other approaches, connectionism and embodied cognition. Although there are few scientists who claim that these approaches could replace the classical symbolic approach, we argue here that these approaches can complement the classical approach. We acknowledge their importance and show that they can address issues to which the classical approach does not give enough attention.

As elaborated in section 4.5, in order to create a cognitive plausible and a *social* Multi-Agent System, ACT-R needs a task environment that can serve many ACT-R actors. This should be possible without many changes to ACT-R itself. What is needed is a communication interface that allows ACT-R to be plugged into a MAS.

The most important problem that we encountered during the implementation of ACT-R in a social setting was the absence of social mechanisms or representations that allowed for social interaction and coordination. The absence of these mechanisms is not surprising given that ACT-R's interests rest in the domain of the cognitive psychologist.

We are interested in simulations that allow us to study organisations at different levels, the organisational level—interaction between groups or organisations, the social level—interactions between individuals and/or groups, and the individual level—explanation of behaviour of the individual that depends on internal cognitive mechanisms.

In order to explain behaviour at the individual level, we adopted ACT-R. However, in the absence of social mechanisms and social representations in ACT-R, there was a need to draw on other theories. Therefore, we adopted social constructivism and semiotics, as explained in chapter 3, to enrich ACT-R with social constructs/representations, and social mechanisms. This provided us with a stronger explanatory power as regards behaviour at the social and organisational level.

In the next chapter, chapter 5, we explain the design and implementation issues of a new cognitive architecture RBot. The new architecture RBot derives its cognitive plausibility, design and cognitive theories mainly from ACT-R. However, ACT-R is in the first place a cognitive architecture that emphasises cognition rather than social interaction. Therefore, we implement new social mechanisms and social representations allowing for social or normative interaction. In the same chapter, we develop a task environment (MRS) that is able to support

interaction for many RBot actors. RBot actors are plugged into the task environment to allow the RBot actor to perceive signals and signs from other RBot actors. Thus, the next chapter is the description of a design based on a combination of chapter 2—Multi-Agent Systems, chapter 3—the social actor, and chapter 4—the cognitive actor.

CHAPTER 5

RBot: Architecture and Design

In the previous chapters we have given the foundations for the design of a new cognitive agent-based computational social simulation model. In this chapter, we will explain the design of a new cognitive architecture RBot, an object-oriented architecture that can easily be plugged-in into a multi-agent platform. In this chapter, we will design such a platform or task environment as well; our so-called Multi-RBot System (MRS).

However, we will first express again the need for such a new actor[1] architecture. Hence, the first question for the reader should be: why developing another multi-actor simulation model and why not using existing software toolkits that give the possibility to model actors?

Although the problem of finding multi-agent toolkits is no problem (there are many available[2]), many agent toolkits are developed with specific areas of research in mind. Such toolkits model for example social phenomena and are focused on the overall behaviour of populations (large populations), waiting queues and bottleneck studies. Or they focus on the intelligent agent itself and when that is the case, they focus only on certain aspects of the agent. Even when they focus on many aspects of the individual interacting with the environment, the assumptions of the model are most of the time not built upon findings in cognitive psychology (such as ACT-R does). Therefore, an initiative was taken to develop a new cognitive architecture from scratch, of course with other existing agent/actor technologies in mind.

This chapter will follow a structure that starts with an overview of the requirements of the architecture and these requirements will guide the design-process of our architecture. Next, a section will follow that elaborates the gen-

[1] In this chapter we will use the term actor, because we are of the opinion that our agent is more autonomous and 'similar' to a human actor than for example a web-agent.

[2] See http://www.agentlink.org and http://www.multiagent.com.

eral architectural design of the complete system, i.e. what architectural patterns are applied. Following up this section, we apply a bottom-up approach in designing the system. We will first start a design process of a single actor and then go into details of how we will unite multiple actors into a multi-actor system.

The chapter will end with a short discussion that functions as an introduction to the next chapter. That chapter will discuss demonstrative experiments and the configuration of the actors and the task environment in detail.

5.1 General Requirements and Overview of the Architecture

In this section, we will first give the general requirements that are necessary for our architecture. Next, we will give a short explanation of patterns and then choose some of those patterns for the general design of our architecture.

The general requirements depend on the research questions we have posed in chapter 1. We will make use of the research questions (1—1.1 until 1.3) and the chapters about MAS, the social and the cognitive actor that deliver detailed requirements for the design of our RBot architecture. The research questions express three general requirements that the newly designed architecture should fulfil:

1. Research question 1.1 indicates that the system should support social behaviour and that such behaviour should be explained in terms of individuals and their interrelations. Distributed systems that are applied in DAI are suitable candidates for modelling such a system. The latest development in the area of distributed systems is the Multi-Agent System (see chapter 2), which is an appropriate methodology for modelling entities that interact with each other and the environment.

2. Chapter 3 answered research question 1.2 and discussed what was necessary for an actor in a MAS to exhibit social behaviour. The discussion made clear that the social construct is a (shared) unit of knowledge that can guide social behaviour. The social construct as a representation in the mind of the actor can be expressed as a data structure in the mind of the actor. Therefore, an actor needs a mind where representations can find their place.

3. Research question 1.3 addresses this issue, and more specifically, what kind of actor can handle representations, signs and social constructs. Chapter 4 explained the cognitive actor and stated that the ACT-R architecture can very well serve as an architecture for our cognitive plausible actor. Therefore, the requirement of our cognitive architecture is to inherit the model, the characteristics and the cognitive mechanisms of ACT-R in order to design a cognitive plausible actor.

The design of an architecture needs a methodology that supports proper design. Patterns provide solutions and methods for solving problems that often reoccur. In this chapter, we will adopt patterns and apply them for finding an

appropriate design that suffices our requirements brought forward by our research questions. In the next section, we will first explain shortly what patterns are and with help of those patterns, we will create an abstract design of our cognitive agent-based social simulation system.

5.1.1 Patterns

Patterns are reusable solutions for problems that occur during software development. Experienced designers (or programmers) find themselves often in situations or problems that have occurred before. In our case of designing a MAS, patterns can provide solutions for the design of our architecture. But what is exactly a pattern?

Christopher Alexander defined a pattern as follows:

> Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution...[It] describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over... (Alexander, Ishikawa, Silverstein, Jacobson, & Angel, 1977, p. x)

There are many classes of software patterns, i.e. architectural patterns, analysis patterns, design patterns, creational patterns, structural patterns, and so on, from which design patterns are the most well known (cf. Gamma, Helm, Johnson, & Vlissides, 1995). Three conceptual levels of patterns can be distinguished by categorising patterns into architectural patterns, design patterns and idioms (or coding patterns) (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996).

- An *Architectural Pattern* expresses a fundamental structural organization or schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.
- A *Design Pattern* provides a scheme for refining the components of a software system, or the relationships between them. It describes a commonly recurring structure of communicating components that solves a general design problem within a particular context.
- An *Idiom* is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language.

In this chapter, we want to elaborate the architectural aspects and not a complete technical description of the architecture. We will adopt architectural patterns to describe the design of the architecture. The architectural patterns can be classified according to architectural views:

> An *Architectural View* is a representation of a system from the perspective of a related set of *concerns* (e.g. a concern in a distributed

> system is how the software components are allocated to network nodes). This representation is comprised of a set of system *elements* and the *relationships* associated with them (Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, & Stafford, 2002). An *Architectural Pattern*, on the other hand, defines *types* of elements and relationships that work together in order to solve a particular problem from some perspective. (Zdun & Avgeriou, 2005, p. 4)

In designing our architecture in the next section, we will first select the views that match the concern. According to these views, we will select the architectural patterns that provide the best solution for our problem.

5.1.2 Overview of the architecture

The overview of the architecture has been designed based on a combination of a set of general architectural patterns. We will follow the design of a MAS (requirement 1), which requires that the components of the architecture should have a high degree of *autonomy*. The Component Interaction View (CIV) (Zdun & Avgeriou, 2005, p. 28) contains individual components that interact with each other but retain their *autonomy*, i.e. they merely exchange data but do not directly control each other. Interaction can be performed synchronously or asynchronously and can be message-based or through direct calls. These components can be distributed. Therefore, the CIV is closely connected to the Distributed Communication View (DCV). The CIV and the DCV are suitable to be applied for designing a MAS, because of their characteristics of autonomy and distribution, respectively. The overview[3] of patterns associated with CIV and DCV is displayed in figure 5.1.

In most multi-agent simulation systems, autonomous actors 'live' in a task environment, which is a separate process that is inaccessible but can be influenced by the actors. Hence, in the design of our architecture, we have made the decision to separate the task environment and the actors. Because the task environment (supplier of the environment) and the couplings with the actors (clients: consumers of the environment) are known at design time, the Explicit Invocation[4] pattern is a general architectural pattern that can be applied for our architecture. Within the Explicit Invocation we can make a decision between the Client-Server (CS) pattern and the Peer-to-Peer (PtP) pattern. The CS pattern consists of a client and a server, two independent components that are not equal to each other: one, the client, is requesting a service from the other, the server. On the other hand, the PtP pattern is somewhat similar to the CS pattern, however all components are equal and can provide as well as consume services. The pattern applied in our design is the Client-Server pattern, i.e. the components client and server are not equal; the server is the task environment and the actors are the clients.

[3] For a complete view, see Zdun and Avgeriou (2005).

[4] In the Implicit Invocation pattern, the invocation is not performed explicitly from client to supplier, but indirectly through a special mechanism (Zdun & Avgeriou, 2005, p. 31).

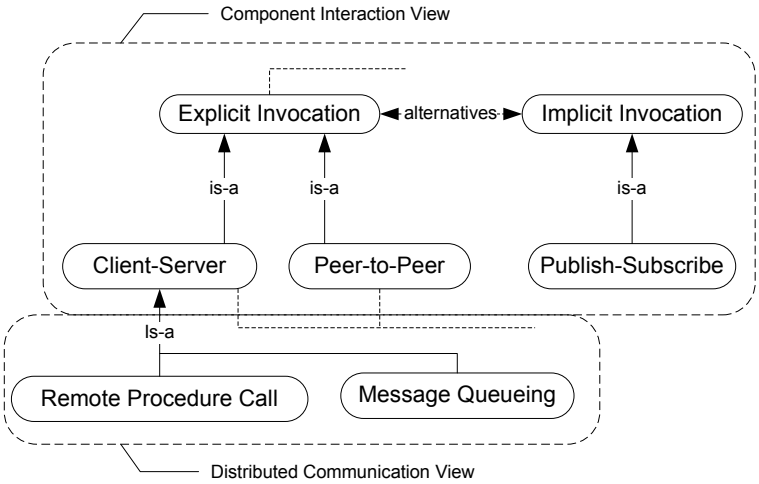


Figure 5.1: Patterns for component interaction and distribution (adapted from Zdun & Avgeriou, 2005).

Apart from a server that delivers the task environment, we also want to implement a backend server that has the function of storing simulation data that are produced by the actors. In that case, results can easily be interpreted after a simulation did run successfully. Hence, when the simulation runs, the actors store their experiences in the backend server. After a run of a simulation, the server changes its functionality and becomes a client that retrieves data from the backend server and presents data or graphs to the user of the simulation program. The combined design delivers a *3-Tier-architecture* (called Multi-RBot System or MRS) that is displayed in the following figure.

The 3 components that can be discerned are the following:

1. The actor: the actor is an autonomous cognitive plausible actor consisting of a production system called RBot which is similar to ACT-R and is able to register itself with the task environment and thereby able to perceive the physical environment and interact with other actors. The actor is the most complex component of the complete architecture and will be discussed in section 5.2.

2. The backend server: the function of the backend server is to store experiences that each actor encounters during its lifetime in a simulation run. This varies from sub-symbolic data (activation), symbolic data (chunks) and physical data, such as place, time etcetera. The database functionality will be discussed in section 5.3.

3. Mixed tier (server-client): a component that serves three functions: the first function, in the function of server, is taking care of control, registration, synchronisation and communication between actors. The second function is presenting a virtual (simulated) world and supplying a task environment—a physical and communication environment—for actors to

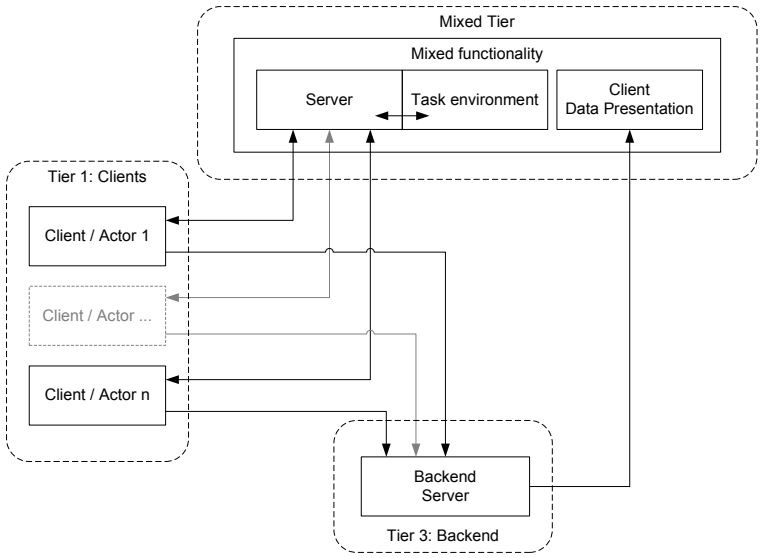


Figure 5.2: Tier client-server architecture; Multi-RBot System (MRS).

live in, i.e. it enables actors to perceive each other and communicate with each other. The user can configure the input parameters of the task environment (e.g. number of actors allowed in the simulation) and the process (e.g. start, pause and resume the simulation) of the simulation. The third function, as a client, is concerned with the presentation of output data generated by the simulation that has accumulated in the backend server, i.e. as a client, the component is able to process output data in order to present them in graphs or output-files to the end-user, ready for interpretation. We will elaborate the mixed-tier component in section 5.4.

The Client-Server pattern seems to be an appropriate pattern for our architecture. However, in a Client-Server architecture, the client has a direct connection with the server and when it makes a request to the server, the server is a passive component that only responds to its environment and is *not* pro-active.

In our architecture, we require a more active component for the server than the CS pattern describes, i.e. the server controls synchronisation between the agents by sending time-step events to the agents. Therefore, we adopt the Publish-Subscribe (PS) pattern (see figure 5.1) to complement our architecture.

> [The Publish-Subscribe pattern] allows event consumers (subscribers[/actors]) to register for events, and event producers to publish (raise) specific events that reach a specified number of consumers. The [PS] mechanism is triggered by the event producers and automatically executes a callback-operation to the event consumers. The mechanism thus takes care of decoupling producers and consumers by transmitting events between them. (Zdun & Avgeriou, 2005, p. 35)

The PS pattern allows for more autonomy for the agents by a decoupling mechanism, i.e. the PS pattern creates a "bridge between the asynchronous network events and the synchronous processing model of the server" (Zdun & Avgeriou, 2005, p. 36). In our architecture, we want actors to subscribe or register themselves with the server, which is a publisher of task environment events. These events can vary from informing of time-progress, or events that happen in the environment, e.g. change of objects in the field are announced to actors that are in the vicinity of these objects.

Another issue we want to discuss in the overview of the architecture is the communication environment, i.e. how do actors communicate (publish and consume events or messages) with each other with help of the server. The most common communication patterns in the Distributed Communication View are the Remote Procedure Call (RPC) and the Message Queueing (MQ) (Zdun & Avgeriou, 2005, pp. 38–39).

The choice for our architecture is to prefer Message Queuing above RPC. The main reason is that we want to have a communication channel that is reliable even when neither the network nor the receiver is reliable. "The simplicity of RPC's is that they're synchronous; a call happens all at once, the caller blocking while the receiver processes. But this is also the shortcoming of an RPC; if anything goes wrong, the whole thing fails" (Woolf & Brown, 2002, p. 4). Therefore, we "[u]se *messaging* to make intra-program communication reliable, even when the network and the receiver program cannot be relied upon" (Woolf & Brown, 2002, p. 4).

The last aspect we want to discuss is the Interaction Decoupling View that "shows how the interactions in a system can be decoupled, for instance, how to decouple user interface logic from application logic and data" (Zdun & Avgeriou, 2005, p. 24). In our 3-Tier architecture, all three components (actor, server and backend) are equipped with a separation of model, view and control, i.e. a Model-View-Controller (MVC) pattern. The MVC divides the system into three different parts:

> [A] *Model* that encapsulates some application data and the logic that manipulates that data, independently of the user interfaces; one or multiple *Views* that display a specific portion of the data to the user; a *Controller* associated with each View that receives user input and translates it into a request to the Model. View and Controllers constitute the user interface. The users interact strictly through the Views and their Controllers, independently of the Model, which in turn notifies all different user interfaces about updates. (Zdun & Avgeriou, 2005, p. 25)

Figure 5.3 shows an example of a Model-View-Controller pattern.

The main advantage of separating model from view and controller is that one of these three components can change (e.g. adding/changing a view) without affecting the structure of the others.

A similar advantage is the decoupling of actors and environment. In the first place, we can run a simulation with a task environment, i.e. the actor 'lives' in a

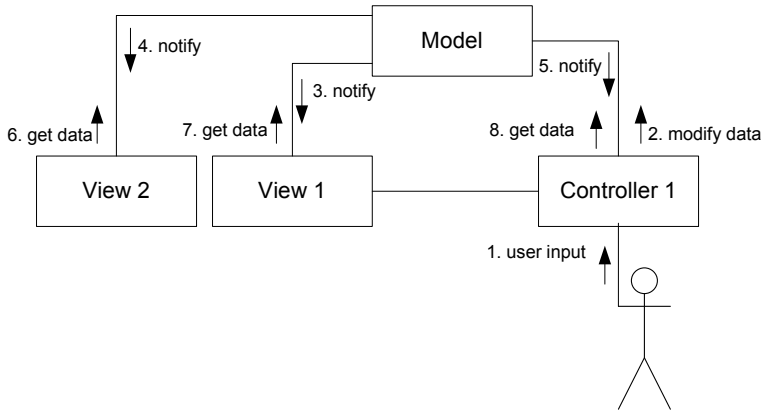


Figure 5.3: Example of a MVC pattern (Zdun & Avgeriou, 2005).

task environment and communicates with other actors—the MRS-mode (Multi-RBot System). On the other hand, it is possible to run a single actor and perform a stand-alone simulation. The *stand-alone* simulation is similar to the ACT-R simulation.

In the next chapter, the first experiment shows a RBot stand-alone simulation in which the actor performs a cognitive arithmetic calculation of adding numbers. The second and third experiment are configurations of two RBot actors that are situated in a task environment (MRS-mode). The remainder of the chapter will start with the design of the cognitive plausible actor architecture, i.e. the stand-alone RBot actor. Next we will discuss the other components—the database and the task environment—that together with RBot form the MRS (Multi-RBot System).

5.2 The cognitive actor architecture

In this section we want to discuss the design of the cognitive architecture. The cognitive plausible actor/cognitive architecture is the most elaborate component of our complete software architecture. The architecture is autonomous, in the sense that it can function independently and can be integrated with any Multi-Agent System, varying from an environment with web-based agents, a robotics environment or a simulation environment as in our case. Apart from autonomy, we want to state that a cognitive actor requires a physical symbol system in order to exhibit intelligent action (Newell, 1980). Such a physical symbol system can be implemented with help of a production system.

From a system architecture perspective, production systems like our architecture RBot, ACT-R and SOAR are partly interpreters, and more specifically: they are rule-based systems, which is a *pattern* with the following properties:

> [A Rule-Based System] consists of mainly three things: facts, rules, and an engine that acts on them. Rules represent knowledge in form of a condition and associated actions. Facts represent data. A

> Rule-Based System applies its rules to the known facts. The actions of a rule might assert new facts, which, in turn trigger other rules. (Zdun & Avgeriou, 2005, p. 24)

RBot is a rule-based system, however, just like ACT-R, it is more than just a rule-based system: it is a cognitive architecture and comprises among the rule-based system cognitive mechanisms that are based on cognitive theories as explained in chapter 4.

The structure of this section will discuss the components of the actor architecture RBot, i.e. the memory model and components: the declarative, procedural and working memory (goal-list and goal stack), the perception component, the physical memory (awareness) component, the message memory component, the social construct memory component and the cognitive engine.

The actor's architecture (figure 5.4) resembles the model of ACT-R version 4 (Anderson & Lebiere, 1998) and version 5 (Anderson et al., 2004). The model is the result from research in software development during which we developed a memory model and a production system based on ACT-R, i.e. the RBot project (cf. Roest, 2004).

If we compare the model of RBot with the model of ACT-R in the previous chapter, then there are a couple of differences. First the access to the memory (memorymap), discussed in the next section, is different in comparison with ACT-R. Next, the function of the goal stack, the cognitive engine[5] (matching, conflict resolution and execution), the declarative, procedural and perception memorymap are similar to the model of ACT-R which is discussed in the previous chapter. The difference with ACT-R is the addition of a goal feeder, the social construct memorymap, the message memorymap and the physical memorymap. The function of the goal feeder is to store goals that need to be processed, i.e. it is possible to equip the actor with a plan of goals and these goals are delivered one by one to the goal stack for the cognitive engine to be solved. The motor part of RBot consists of a physical memorymap that maintains the current position and time of the actor (physical awareness) and the message memorymap that regulates communication with the outside world. The social construct memorymap is a new feature that maintains (social) knowledge about the role the actor plays in the social environment with other actors. The social construct allows the actor to react instantaneously to social events and enables the actor to respond to perception and messages from other actors received from the outside world. These interactions can for instance adjust sub-symbolic properties, such as the utility of procedures and thereby influencing the execution of procedures. The remainder of the section will explain the different components in detail.

The model and the memory of the actor are organised in a similar structure as ACT-R, however the way the model is implemented in JAVA is quite different. Not only the language chosen to implement the model is different, also the design of the memory and its components is quite different. Because the model

[5] Although the human brain does not contain a cognitive engine, we refer in this dissertation to it as a reasoning unit that controls the operations on symbols or representations in the mind of the actor.

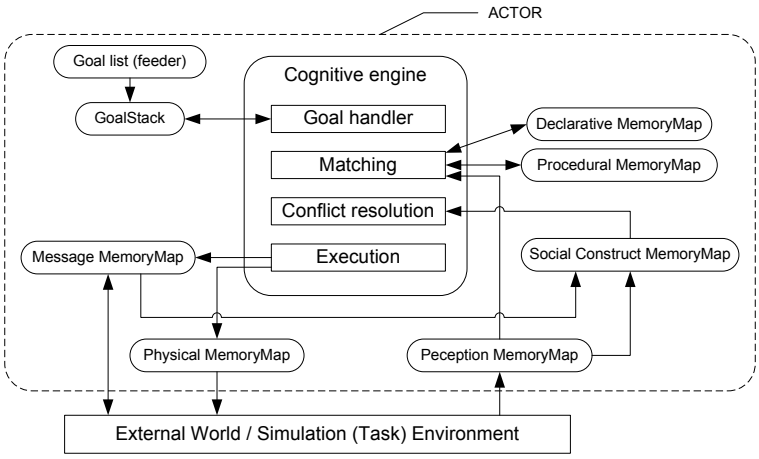


Figure 5.4: RBot; comparable with ACT-R (see previous chapter).

is implemented from theory, and not from the original LISP code, the model may show differences. However, the way the model can be used in practise is similar to ACT-R.

Building up the model from scratch towards a multi-actor simulation model needs a clear explanation about the implementation of the model and its environment.

The cognitive actor architecture (the client) can be divided in three main components: the memory model, controller and handlers, and the user interface, similar to the MVC pattern as discussed in the previous section. The memory model (next section) together with the controller and handlers (section 5.2.2) form the core of the RBot architecture and will be explained in detail.

Besides that, every actor that is part of the multi-actor simulation has its private user interface. This interface (section 5.2.3) allows for configuration and control of the single actor. The user interface of the task environment is a different user interface and is part of the mixed-tier, which we will discuss in section 5.4.

5.2.1 The Memory

The model of the memory arranges information inside the actor's mind. The importance of this can be seen when we for example see the arrangement of the cupboards and the refrigerator in a kitchen. For instance, the moment we feel the need for a cold drink, we tend to go to the refrigerator to get it, i.e. we know from the refrigerator that it is capable of holding drinks and keep them cold; and in our mind, we label the refrigerator as having this capability. Similarly, in the memory of an actor, the indexing and labelling of information (cf. tagging: Holland, 1995) are necessary for the ordering of information and retrieval of information at low cost.

The memory model manages the state and the access of the memory. In a

production system, like RBot, productions and facts are stored in memory in order to be retrieved, changed or added, which is caused by state changes of the actor. The results of these state changes have to be stored in an orderly fashion, allowing easy access and management of the memory.

The memory of RBot consists of components: *chunks*, the generic components of storing information or knowledge in memory that are linked by *links* in a network-like structure forming a graph structure. Chunks contain atomic components, namely slots that can contain objects or data. Besides the graph structure, the formatting (the number and type of slots in a chunk) is defined by its *chunktype (its tag)*. Hence, apart from creating structure in memory, it is possible to classify chunks as well, based on their chunktype.

RBot as well as ACT-R model separate parts in memory to store information (e.g. procedural and declarative memory). Secondly, in both architectures, the components themselves are classified as well (tagging: chunktype). Thirdly, ACT-R and RBot give importance to chunks by ranking them based on activation level (explained later on). Besides some overlap in the structure between RBot and ACT-R, RBot has an explicit graph structure[6] (comparable to the production-graph structure of SOAR) that is a set of vertices (chunks) connected by edges (links). In this way, a network of chunks connected by links can form a semantic network. In RBot, the memory holds a collection of memorymaps. A memorymap, for instance the declarative memory, is a map that holds a graph of chunks and links. The organising of graphs in different maps gives the possibility to assign different functionality to those maps (for instance, perception memorymap connects to functions that relate to perception). The way this is done in RBot is slightly different and will be explained in the next section.

5.2.1.1 Memorymap, buffer and stack

The design of the memory follows closely the design of ACT-R in describing the components. We also define three main memory modules: the goal stack, declarative and procedural memory. In RBot, these main memory modules are special cases of a generalised memory organisation based on memorymaps.

A *memorymap* is a container of memory components that gives the actor access to these memory components. Whereas ACT-R applies a functional division of the memory in modules, the memorymap in RBot is a better and more flexible approach to structure the memory. The memorymap creates access to chunks in memory by defining entrance chunks, see figure 5.7.

Chunks can be defined as part of different memorymaps in memory. For example, a chunk can be part of the declarative memory, but at the same time can also be part of the procedural memory, without being duplicated in the memory. This is possible because the memorymap holds references to objects in memory and forms an object-oriented memory structure. In other words, chunks are separated in different maps, but they are not that strict separated that they cannot be part of several maps.

[6]Graph structures (implicit graphs/soft links) are also possible in ACT-R by connecting slots in chunks to other chunks.

RBot defines three types of memorymaps (figure 5.5): the *memorymap*, the *buffer* and the *stack*. The memorymap is the super container of the memory that can theoretically contain an infinite number of chunks. Buffer and stack are both based on the memorymap and inherit properties from the memorymap[7].

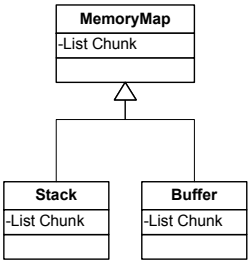


Figure 5.5: Three types of memorymaps.

The buffer has a predefined size and behaves like a FIFO (First In First Out) buffer that allows a buffer to store only a maximum amount of chunks at any time. For example, the visual-buffer could be defined as containing a maximum number of elements: when more items are stored in the visual-buffer, the buffer removes items that have stayed in the buffer the longest.

The stack on the other hand acts like a LIFO (Last In First Out) buffer with infinite size and points at the latest addition of a chunk, i.e. the chunk that is added (pushed) on top of the stack will be removed when there is a remove (pop) request made to get a chunk from the stack.

The generic methods of accessing and controlling a memorymap are adding chunks, retrieving chunks, and searching for chunks based on their properties, e.g. a certain defined chunktype. Figure 5.6 shows an example of a buffer with a capacity of five chunks to demonstrate the way a memorymap (buffer) functions.

In this case the buffer is full, adding another chunk will make the fifth chunk of chunktype X automatically be removed. And for example, when we want to retrieve chunks of this memorymap, we can retrieve all the chunks or chunks that are based on a certain defined chunktype (its label). Thus, the function of the memorymap is: organising memory components and giving access to them according to the properties of the memorymap type (in this case a FIFO buffer).

In RBot, it is possible to define custom memorymaps and add them to memory. Some memorymaps are standardised and have a predefined function. These memorymaps are the following:

- Goal feeder memorymap: manages and schedules the supply of goals to the goal stack.
- Goal stack: the internal stack of the goalhandler that manages sub goaling.

[7]The inheritance principle (Campione, Walrath, & Huml, 2001, p. 52) is an object-oriented property. Hence the memorymap and its children are object-oriented classes.

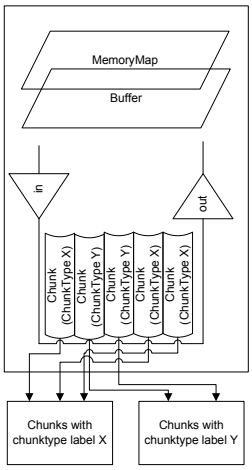


Figure 5.6: Example of a buffer / memorymap.

- Declarative memorymap: has the function of storing chunks that represent facts.
- Procedural memorymap: stores the procedure chunks in memory.
- Social constructs memorymap: storage of social constructs.
- Physical memorymap: storage of physical properties of the actor.
- Message memorymap: messages, in the form of chunks, destined for communication with the outside world are stored here.
- Perception memorymap: messages of perception are stored in this map in the form of chunks.

Every memorymap has the ability to store memory components such as chunks. These components are holders of information and describe states in memory and influence how state changes can take place. The design of these components will be discussed next.

5.2.1.2 Memory Components

The Memory Component is the super-class of all components (e.g. chunks, links) in the memory. Its main function is to administer the activation of the components and making them aware of time progress during the simulation. All components that extend the generic Memory Component have the properties of activation and time awareness at the so called sub-symbolic level.

After creation of a memory component, the component (be it a chunk or link) keeps track of its presentations (the number of times the component has been harvested or merged by the engine), thereby storing the history of its presentations in order to be able to determine its activation value. The calculation

and determination of the current activation of the memory component will be explained in the next section about controllers and handlers.

In this section, we will start with explaining the *chunk*, the *link* and the way those components can be *structured* and *managed*. Next, we will discuss the *procedure*, a component that consists of a structure of links and chunks and the *formatting* of chunks (XML) that enables the actor to translate memory components into computer files or messages that can be sent to other actors. We end this section with the *social construct*; a memory component comprised of a structure of links and chunks that gives the actor the capability of constructing social situations in its mind.

Memory Component :: Chunk

Chunks are the most important memory components in the architecture, they are the building blocks for most other memory components. With help of links (another memory component), chunks can be connected to other chunks and can form complex structures, comparable to molecular structures or neural networks. The chunk is a container that contains *slots*. A slot has a name and holds objects or data. With help of a classifier, the *chunktype*, the number and names of slots in a chunk are defined (tagged), comparable to the DTD[8] of an XML document.

Chunktype Chunktypes are defined in main memory and only chunks that are valid according to a predefined chunktype are accepted as "understood" elements. To compare with XML documents: when XML documents are transferred between computers (or actors), they only can be parsed if they are following a defined format; only chunks that are defined by a known chunktype are parsed by the actor and stored in memory. New chunks of an unknown chunktype can only be absorbed by the actor's memory if the chunktype is given as well or if a separate mechanism exists for storing unknown chunks and allows for the generation of new chunktypes. Chunktypes give restrictions in the way information is stored, but give the ability for indexing and easier search of specific types of information stored in memory.

Memory Component :: Link[9]

The link is a memory component that enables to connect two chunks, thus creating the possibility to construct a graph or network structure. The advantage of links is that they order memory by linking certain chunks that are semantically related to each other.

For instance, connect chunk5 (value 5) together with chunk7 (value 7) and chunk12 (value 12). In this case, an addition or subtraction of two chunks would result in a faster retrieval from memory because they are directly connected to

[8] DTD (Document Type Definition) defines the legal building blocks of an XML document.

[9] In the current RBot architecture, the activation levels of links are not used, but its availability can be used for future algorithms.

each other[10]. In the definition of RBot, the link can contain two chunks only, while the chunk can be related to an unlimited number of links. The current implementation of the link is bidirectional, i.e. the moment a link is constructed, both chunks have a link to each other. With the help of the chunktype, chunks can already be sorted, but the addition of links between chunks gives the possibility of creating very complex structures such as trees, or other types of graph structures that order chunks even more, e.g. a semantic network.

Memory Component :: Structure

With help of links, chunks and a memorymap, it is possible to create ordered (complex) structures of chunks in a memorymap. Figure 5.7 shows an example of a created memory structure. The entrance chunks form the elements that are directly accessible from outside the memorymap and are explicitly added to the memorymap. All other elements can only be reached indirectly, by following a specific link towards them.

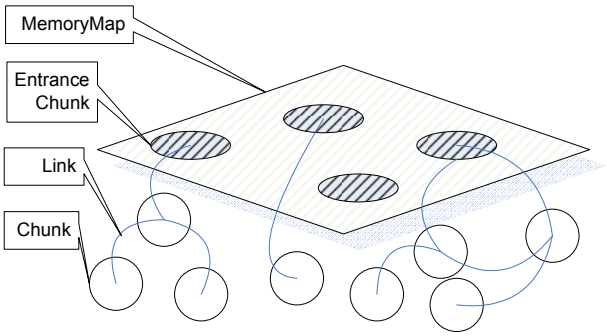


Figure 5.7: Example of a memory structure.

The purpose of the entrance chunks is to create a separate access layer in the memorymap. The advantage of such a layer is that it creates faster access and sorting of chunks that are the most important.

The structure of components in the memorymap and the relations between its components can be explained with help of an UML[11] diagram, see figure 5.8.

The class diagram shows that an actor has a memory that exists out of one or more memorymaps. Apart from that, the memory also stores the chunktypes that are known to the actor. The memorymap, as discussed before, can be a simple container (List) of chunks, a FIFO buffer or a LIFO stack. The ChunkType defines the structure of the Chunk. Both, the Chunk and ChunkType, are constructed out of Slot components. The Link and the Chunk are memory components that inherit properties and methods from the MemoryComponent, i.e. Chunk, Link and ProcedureChunk (explained later) inherit properties, such as

[10] If the three are connected with many other chunks, then it can result in slower retrievals according to the fan-effect. See previous chapter and Anderson (1974).

[11] Unified Modeling Language (cf. Booch, Rumbaugh, & Jacobson, 1998).

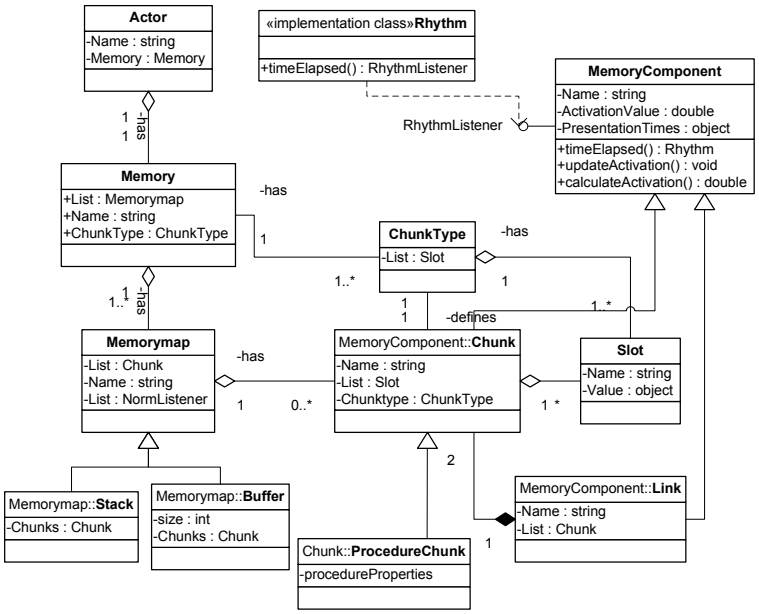


Figure 5.8: Memory components class diagram.

the field ActivationValue and the method updateActivation() from the MemoryComponent. The MemoryComponent has a listener that responds to time-based events generated by the simulation clock (the Rhythm).

With this set of memory components, it is possible to create complex graph structures, e.g. a semantically linked network structure. Consequently, a memory of memorymaps with chunks, interconnected by links and holding slots formatted by the chunktype, can be defined as part of the actor.

Memory Component :: Management

In order to maintain the memory, RBot defines functions for managing memory: adding or creating, moving and removing, and searching for chunks and links. Chunks can be created by internal cognition or by encoding of signs perceived from the environment.

The chunk can be added to the memorymap in two ways: adding chunks directly to the memorymap as an entrance chunk or adding by linking the chunk to another chunk that is already part of a memorymap, i.e. in the case of procedure chunks, a new constraint can be added to a condition side by linking a new chunk to the condition Rule (explained in the next section). In the case of a link, it can be added by specifying the two chunks to which it needs to be attached.

Removing any memory component is exactly the opposite of adding a component and moving a component is a combination of first removing and then adding.

Indexing (classification by index, i.e. tagging/chunktype) memory gives the ability to create specific algorithms for memory-search because knowledge is

classified by chunktype, linked to other chunks and member of a certain memorymap.

Algorithms to search knowledge can vary and many possibilities to search memory are possible. The singleton Algorithm class (discussed in the handler section) is a handler that handles all complex searches for information in memory and is separated from the memory components and memorymap. The separation of search-algorithms from memory objects has the advantage of easier and better maintenance and reusability.

Multiple references to chunks In RBot, there is no restriction in having double or more references to chunks; chunks can be member of many memorymaps at the same time. The memorymap is used to create different parts in memory that are designated a different functionality, i.e. the same chunks can become member of different memorymaps. In this case, the chunk is still unique, but there are more ways of reaching the chunk by reference in memory.

The remaining issues regarding management are more related to processes than to structure. Hence, we refer to the next section that elaborates controllers and handlers.

MemoryComponent :: Procedure, Rule, Demand and variable chunks[12]

Both ACT-R and RBot distinguish different cognitive components regarding the declarative memory and the procedural memory. Whereas the declarative memory contains components that describe states and facts in memory, the procedural memory contains components that describes state *changes*.

The difference between ACT-R and RBot is the design of the software components. In ACT-R, a separate component is created for defining a procedure, whereas in RBot, the procedure is a small network of chunks that contain information about the procedure.

In RBot, a production or procedure is a small network with a procedure chunk as root, to which *rule chunks* are linked. A production normally exists out of two sides, a condition and an action side. The main function of the rule chunk is to make clear to which side constraints (or demands) belong and to which memorymap that condition or action refers to, e.g. as goal stack condition, declarative memory condition, goal action and so on.

The rule chunks are linked to *demand chunks*. When the demand (or pattern[13]) chunk is connected to a condition rule chunk (that refers to a certain memorymap), then the demand chunk states to which pattern the chunks in the referenced memorymap have to be matched to. In the case of an action rule chunk, the demand chunk will be processed by the cognitive engine that can result in a change of the contents in the referenced memorymap.

The role of variables in a procedure is to allow for procedures containing patterns that match to more than only one situation. The influence of variables

[12] During development, these names (rule, demand and so on) evolved. We see a production as a combination of condition-action rules. A production consists out of a set of condition and action rules that apply to that production.

[13] A pattern chunk is a chunk that contains variables and describes which other chunks do match with this chunk (pattern-matching).

will be discussed in the section that discusses generalisation and specialisation of procedure chunks.

An additional feature of RBot compared to ACT-R is that RBot allows for variables in the name of a slot in a demand chunk. These variables are expressed in a separate DSN (Dynamic Slot Name) chunk, a holder of slotname variables that is linked to the demand chunks. The DSN chunk will also be explained in the section about generalisation and specialisation. The complete structure of the procedure is shown in figure 5.9.

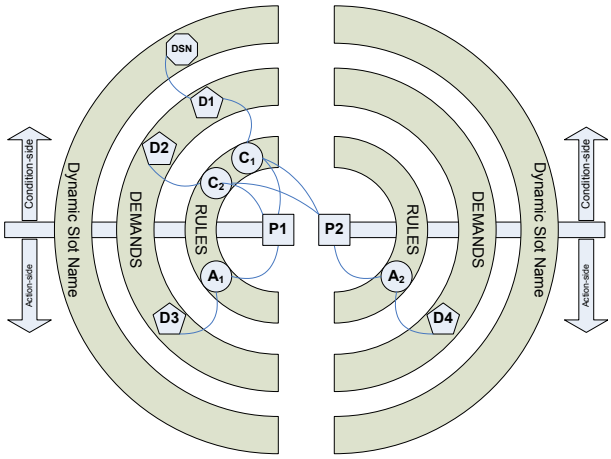


Figure 5.9: Structure of the procedure.

The condition and action chunks are defined by the Rule-chunktype that exists out of three slots: the type ('part'), modus and referenced memorymap. The type defines if the chunk is a condition or action rule chunk, the "modus", expressed by a '-', '+' or '=', determines what type of operation on the referenced memorymap, the third slot, is taking place[14]. In the current implementation of RBot, there is only one condition modus possible, the '=' sign, which stands for compare; comparing in the sense of a match between the condition of the procedure and the goal that is currently in focus or chunks that have to be retrieved. The action side on the other hand has more possibilities, the '=' is the action of modifying, the '-' for removing and the '+' for adding chunks. The demand chunk defines the specific request to the condition or action chunk and determines what slots of what type of chunk the operation has effect on.

The advantage of splitting the procedure into different parts is shown when adding another procedure P2 that reacts on similar conditions, but has a different action A2 (see figure 5.9). Procedures can be interlinked when they possess similar conditions, actions or demand chunks. By creating linkages, memory can be saved and managing the procedures can be done more efficiently.

Inside a part (the condition or action part), the specific constraints or demands to the conditions and actions can be set. Procedures can only be activated

[14]We follow as much as possible the conventions of ACT-R.

when their condition side matches with the current state of the goal. Hence, the procedure always contains a demand chunk that refers to the goal stack, otherwise it will never respond to goals.

Generalisation vs. specialisation of procedure chunks In the previous chapter, we explained learning mechanisms at the symbolic level. Anderson and Lebiere (1998, p. 106) distinguished discrimination, generalization, composition and proceduralization. Procedures that are created by such a learning mechanism can either become more general or more specific applicable to the context in which they operate. We define two processes: generalisation (Anderson's generalization and proceduralization) and specialisation (Anderson's discrimination and composition). We can infer from these processes that there is a variation and change in the structure of procedures as a result from generalisation and specialisation, varying from more general towards more specific procedures.

General procedures are applicable to situations in which a procedure needs to solve more than solely one goal (or one specific retrieval). How general a procedure is, depends on the number of condition chunks and the number of variables that are available in the demand chunks and DSN chunks linked to those demand chunks. Fewer condition chunks, defines fewer requirements that are necessary to match the condition of the procedure, and therefore the procedure is more general applicable.

The strength of variables is that it allows for defining procedures that are able to react on more than one situation. Thus, the more variables defined in a condition-rule (e.g. the goal-condition) the more situations are applicable for the procedure to react on.

A variable is expressed by an '=' in front of the variable name, e.g. =num1 (similar to ACT-R). The variable gives the ability to transport a value from the condition side of a production to the action side. The more variables in the demand chunks at the condition side, the wider the range of goals the production can react on, the more generalised the procedure. Compare for example the following demand chunks[15] that react on a goal of adding two numbers.

The goal chunk is of chunktype addition_problem with two slots known (number1 == 2 and number2 == 3) and the value of the answer is unknown. The condition demand chunks are chunks that are to be matched with the goal chunk. Both chunks are responding to the goal chunk, however, demand chunk 1 has two variables and also responds to a goal chunk with for example number1 = 4. Demand chunk 2 is more specific and only responds in cases where number1 equals 2. Thus, demand chunk 2 is more specific than demand chunk 1.

The case of generalisation can go one step further by introducing more places in the procedure where variables can be substituted. The open production issue, discussed in the ACT-R version 6 Proposals (Bothell, Byrne, Lebiere, & Taatgen, 2003), suggests to create variables in the chunktype and the slot name (which

[15]We apply XML (eXtensible Markup Language) notation for our chunks. See http://www.w3.org/XML

Goal chunk

```
<chunk name="addition" type="addition_problem"/>
     <slot name="number1" data="2"/>
     <slot name="number2" data="3"/>
     <slot name="answer" data=""/>
</chunk>
```

Condition Demand chunk 1

```
<chunk name="addend1" type="addition_problem"/>
     <slot name="number1" data="=num1"/>
     <slot name="number2" data="=num2"/>
</chunk>
```

Condition Demand chunk 2

```
<chunk name="addend2" type="addition_problemv/>
     <slot name="number1" data="2"/>
     <slot name="number2" data="=num2"/>
</chunk>
```

could have some implications at the theory level of ACT-R). RBot does not implement the variable chunktype, but does implement variable slot names (DSN chunks) in procedures and can be seen as an important innovation. The reason of implementation is that during development of complex problems more generalisation in procedures proves to be necessary. The more generalisation, the more abstract the procedures, and the less procedures are necessary to describe a problem space. The following example shows a possibility of applying such a generalisation[16].

Condition: Dynamic Slot Name (DSN) chunk

```
<chunk name="dyn1" type="dsn"/>
     <slot name="number<=num1>" data="=num1"/>
</chunk>
```

The condition states that the production reacts to all goal chunks that have their slot name [number<N>] matching their data value N, e.g. one chunk that matches is the following goal chunk.

Goal chunk 2

```
<chunk name="addition2" type="addition_problem"/>
     <slot name="number4" data="4"/>
</chunk>
```

With this additional property, it is possible to write procedures that are applicable for a larger domain of goals.

Sub-symbolic level The root chunk (e.g. P1 and P2) also functions as a holder for data at the sub-symbolic level. All sub-symbolic data is stored inside

[16]The purpose of the example is only to show that it creates a more general applicable condition chunk when a slot name can hold variables, i.e. it is not directly applicable to solving an addition problem.

slots that are defined by the procedure chunktype. It serves as the holder for efforts, successes and failures, and therefore represents the sub-symbolic level of procedures. This is done by purpose in order to configure easily new sub-symbolic data holders for flexible implementation extensions for procedures in future releases. More details about the sub-symbolic level are found in the next section about controllers and handlers.

Memory Component :: Formatting components in XML

In the discussion about the chunktype, the introduction of the DTD and XML was already mentioned. XML stands for eXtensible Markup Language (Harold, 1999), derived from SGML (ISO 8879), and is a set of rules for defining semantic tags that break a document into parts and identify different parts of the document. XML is an industry wide adopted standard, documented at the W3C (http://www.w3.org/XML), and is already applied in many software applications.

RBot applies XML to translate memory components and memory structures into messages or computer files. This structure gives the possibility for easy transfer between components inside the architecture, storage of components in documents and transfer between actors, or other software applications if necessary. The XML structures in the next table show examples of how a procedure and a goal chunk are modelled in RBot.

Example of Procedure in XML	Example of GoalChunk in XML
<procedurechunk name="StopCounting"> <condition> <rule modus="=" memorymap="goalbuffer"> <chunk name="goal" type="countfrom"> <slot name="Start" data="=num1"/> <slot name="End" data="=num1"/> <slot name="Step" data="counting"/> </chunk> </rule> </condition> <action> <rule modus="-" memorymap="goalbuffer"> <chunk name="goal" type="countfrom"> <slot name="Start" data="=num1"/> <slot name="End" data="=num1"/> <slot name="Step" data="counting"/> </chunk> </rule> </action> </procedurechunk>	<goalchunk name="addition_problem" activation="0.7" type="addition_problem"> <slot name="col1number1" data="5" /> <slot name="col1number2" data="7" /> <slot name="col2number1" data="5" /> <slot name="col2number2" data="7" /> <slot name="col3number1" data="1" /> <slot name="col3number2" data="7" /> <slot name="col4number1" data="5" /> <slot name="col4number2" data="7" /> <slot name="carry" data="start" /> <slot name="nrColumns" data="4" /> <slot name="result" data="" /> </goalchunk>

Table 5.1: Example of a procedure and a goalchunk in XML.

The XML structure is a translation into a kind of knowledge interchange format (KIF) (cf. Genesereth & Ketchpel, 1994) agreed upon by a community of actors and enables actors to exchange and understand complex knowledge structures. The social construct, as discussed in the previous chapter, is an example of a complex knowledge structure that can be exchanged by agents to communicate norms and rules of conduct to other actors.

Memory Component :: Social construct

In a multi-actor simulation, the interaction between actors is important as well, therefore the memory has to hold some information about social constructs. Modelling social constructs is our answer to the research question 1.2, i.e. the social construct as a memory component is a data structure for constructing social situations in the mind of the actor.

As explained in chapter 3, social constructs can be seen as representations of rules for cooperation and coordination, based on intertwined habits and mutual commitments that are often expressed in sign structures such as agreements, contracts and plans. Social constructs are reinforced by their frequent use. Hence, the chunk with its properties of activation and slots is an appropriate container for storing social constructs.

Social constructs can have some or all of the eight properties (as discussed in chapter 3): norms, (un)written, life span, authority, inheritance, scenario, context and roles.

First of all, the social construct has attached norms, which can be implemented by creating a condition-action pair: NormCondition and NormAction. Secondly, in this case the social construct is written and formalised to make it possible to implement it in the architecture[17].

The life span of the social construct in RBot depends on reinforcement of social constructs in memory, i.e. without reinforcement, the actor forgets.

Incoming chunks in perception or other memorymaps are triggers for starting, reinforcing or stopping the life span of the social construct.

Authority depends on the relationship with other actors, for example awareness of other actors. When commands have to be followed up, the actor has to have an ordering of roles in combination with authority.

Inheritance and scenarios are beyond the current implementation and require more work for future releases.

Context dependency is defined by the NormConditions of the norm and they cause an automatic response to situations (context) in which the actor is involved.

The current implementation defines that social constructs are part of the actor and, being chunks, are able to be forgotten or reinforced in time. They are implemented in a way comparable to the subsumption architecture of Brooks and Flynn (1989, p. 2): "The subsumption architecture is a parallel and distributed computation formalism for connecting sensors to actuators in [actors]". This allows the direct activation/de-activation of social constructs and norms based on changes in the perceived context. This activation/de-activation is done more or less parallel to the main cognitive processing. The activation of a norm has as effect that a number of productions get a higher or lower utility as specified in the norm action, and this is done instantaneously. This mechanism for the activation/de-activation of social constructs and the effectuation of the associated norms can be seen as a special case of a general mechanism enabling the cognitive architecture to react directly on changes in specific memory locations

[17]The unwritten social construct is based on interactions (the social level) and will be explained in the second experiment in chapter 6 that discusses the emergence of social constructs and organisation.

(e.g. the perception buffer). The mechanism consists of a listener that listens to changes in the relevant memory locations and an effectuator changing the activation, utility or other sub-symbolic parameters of productions or other chunks.

Concerning the theoretical description of the actor and the social construct theory, we only implement norms that have influence on utility of procedures. The remainder of social norms may require a more complex goal deliberating process that is not implemented yet. Therefore, in the implementation we use conditions of norms that react on changes in memorymaps of RBot, i.e. changes that can occur in parallel, together affecting the utilities in the set of procedures that are responding to the current goal. The current implementation of a social construct is shown in the following figure.

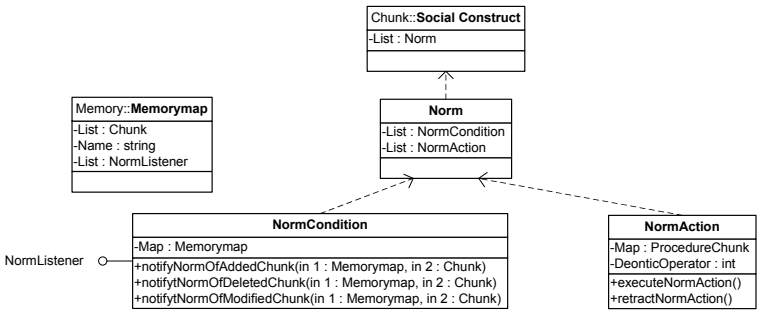


Figure 5.10: Classes that implement the social construct.

As figure 5.10 shows, the social construct contains a list of norms, and the Norm exists out of a NormCondition and a NormAction. The condition contains a NormListener that is connected to a memorymap. The MemoryMap notifies all NormListeners, in this case the NormConditions of all norms when a chunk is added, modified or deleted from the memorymap. The NormCondition filters the chunk and determines if it has to respond to this event and if so, it sets the Norm to active. When the norm is active, it executes the NormAction. The NormAction holds a list of procedures that are influenced by the norm. Thus, the overall effect is that the social construct creates changes in the utility of procedures and thereby promoting or demoting productions in the conflict resolution.

5.2.2 Controllers and handlers

The controllers and handlers section describes the control over the process and the handling of events in this process. In the kitchen example, we can compare the controllers or handlers with the cook in the kitchen who is preparing a meal. The cook first defines a goal for producing food and takes defined measurements of ingredients that are stored in the cupboards and the fridge. The cook transforms the ingredients towards meals according to recipes and either stores the end product in the fridge or serves the product to the restaurant. The cook controls the process of cooking. However many events are handled by ma-

chines over which the cook has no direct control. It is the interface that delivers the control to the cook, and thereby empowers him to indirectly handle events that happen during the process of cooking.

The controllers and handlers of RBot have some similar functionality as the cook. The controller has influence on the input of the recipe (procedures) and the handlers take chunks from the memorymap and transform them and put them back in memory, or send them out the system into messages or movements of the actor. Hence, controllers and handlers have the function of controlling the simulation and causing or responding to state changes in RBot.

A discrete event simulation, like RBot, is based on a time-stepping device that controls the current time and the speed of the time steps (events). Among time dependency, three types of handlers can be distinguished; handlers that respond to time changes, handlers that respond to state changes in memory and handlers that react based on messages from the server. The time dependent handlers are registered with the rhythm controller and act based on time changes. The memory state dependent handlers are registered with a memorymap, and are notified by memory changes and the message based handler reacts on messages sent to the actor. Besides the event-based handlers, there are handlers that take care of a specific job, for example the communication handler that deals with the outside world and the perception handler that ranks perceived chunks in the perception.

5.2.2.1 Goal Feeder and Goal Handler

One important aspect in a goal and procedure oriented architecture is the way how goals are chosen, processed, do fail or succeed. Two handlers in the architecture deal with the scheduling and management of goals; the 'Goal Feeder' and the 'Goal Handler'. The goal feeder is a decision unit that manages the goals that are to be executed, which for example could be done based on a plan. The current implementation is a simple FIFO buffer that handles goals based on arrival in memory. The feeder awaits the moment the goalhandler is not busy and the goal stack is empty, so the feeder can feed the next goal to the goalhandler. The current implementation is simple, but allows to process a number of goals.

The goalhandler is a finite state machine that controls the pattern matching of conditions and firing of procedures, together with credit assignment to procedures and changes in memory. The goalhandler is theoretical comparable to the central production system of ACT-R. However some slight changes have been implemented in the internal handling of the procedures, mainly because the procedures have a different structure. The goal feeder and the goalhandler are shown in figure 5.11.

The figure shows that if the goal feeder contains goals, it is able to feed the goal to the goal stack. The goal feeder is able to feed goals as long as it contains goals and the goal stack is empty. In other cases, the goal feeder has to

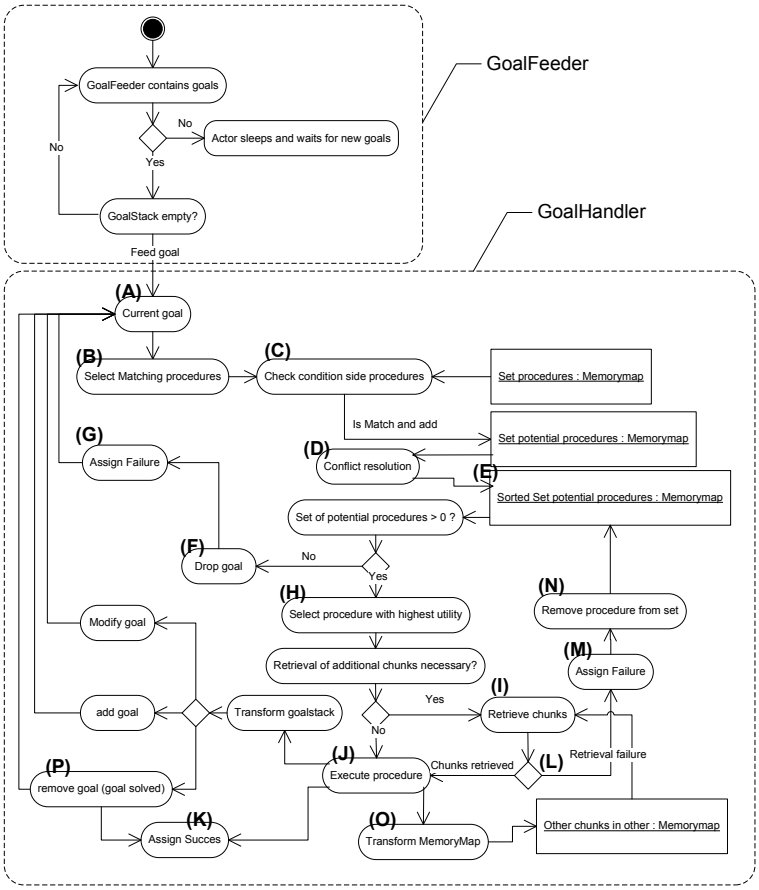


Figure 5.11: GoalFeeder & finite state machine of the GoalHandler.

wait for the goal stack[18] to become empty or for delivery of new goals from the environment (e.g. other actors).

The goalhandler is a kind of finite state machine that uses the goal stack as a working memory of states. The goalhandler has the following cycle:

1. If there is a current goal **(A)** on the goal stack that has focus, then select the procedures **(B)** that match with the current goal. The selection is done by matching the condition of the procedure **(C)** that refers to the current goal. The selection results in a set of potential procedures.

2. In case the number of potential procedures is larger than one, this is seen as a conflict[19]. The next step **(D)** is "conflict resolution", which results in a

[18]The goal stack has a stack pointer that points to the current topmost data item (current goal chunk) on the stack. A push operation decrements the pointer and copies the data to the stack; a pop operation copies data from the stack and then increments the pointer.

[19]If the set consists of one procedure, then that procedure is of course selected and there is no conflict, i.e. during conflict resolution, that procedure is automatically selected.

sorted set of potential procedures **(E)**.

3. In case the set does not contain any procedures at all (no match), then the goalhandler drops the current (sub)goal **(F)**, a failure is credited **(G)** and the goalhandler returns to step 1. Otherwise, if the set is not empty, then the procedure with the highest utility is selected **(H)**.

4. After selection of the procedure, the procedure retrieves chunks **(I)** from memory. The retrieval is only necessary when a retrieval is specified in the condition of the procedure.

5. When there is no retrieval error or no retrieval necessary, the engine will execute **(J)** the action side of the procedure, and the procedure will be credited with a direct success **(K)**. If there is a retrieval error **(L)**, the procedure will be credited with a direct failure **(M)** resulting in the removal of the procedure **(N)** from the set of potential candidates. After removal, the cycle is re-entered by checking if there are still potential procedures left over in step 3 of the cycle.

6. The execution of a procedure can result in a transformation of a goal **(N)** in the goal stack or a transformation in other memorymaps **(O)**. Three types of goal transformation (and transformation of other memorymaps) can take place determined by the action statement of the procedure, i.e. modifying, adding or removing a chunk. A special case is the fact when a goal is removed from the goal stack **(P)**; if a (sub)goal is removed and solved, the procedure(s) involved receive(s) a credit of success **(K)** and the solved goal is added (as a solution) to declarative memory.

7. Return to step 1.

The function of the goalhandler is to manage a number of tasks by outsourcing or delegating them to other objects in the program, i.e. sub-processes that are responsible for a certain task. Delegation of tasks to other objects increases the ease of modification, maintenance and reuse of code. Examples of tasks the goalhandler manages and delegates are:

- Goal stack.
- Pattern matching, e.g. matching a (pattern) chunk in the condition of the procedure with the goal.
- Binding of variables to values.
- Action execution.
- Management of (sub)goals.
- Credit and cost assignment of procedures.
- Administering retrieval errors.
- Managing activation changes.

The goalhandler maintains a goal stack that is responsible for ordering goals at different levels. Goals are modified, added or removed to the stack by the goalhandler, which depends on procedures that add or remove a goal from the stack (or dropping of the goal in case of a failure). The goal feeder is notified by the goalhandler when the goal stack is empty and thereby is ready to receive new goals.

Goal stack

The goal stack is a memorymap that acts as a LIFO buffer and keeps track of a list of goals with on top the goal that is currently in focus. It describes the division of a problem in sub problems by throwing a (sub)goal when a procedure needs to solve a goal that depends on another goal. For example, see figure 5.12, when the main goal is to go from A to B, the sub goal could be to take a step and the summation of steps creates the solution for solving the main problem.

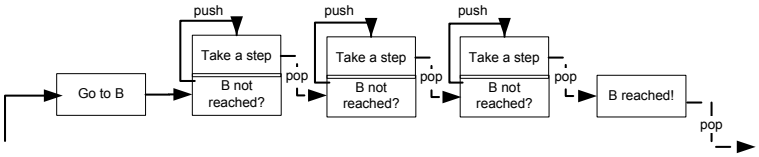


Figure 5.12: Goalstack.

The condition side of the goalhandler

The condition side of the goalhandler mainly deals with pattern matching of chunk(s) in the condition side of a procedure with the current goal in the goal stack. A procedure in RBot is matched in the following way. After selecting a procedure, the goalhandler selects the accompanying rule chunks of the procedure. The rule chunks consist of a mixture of rules containing references to different memorymaps (e.g. goal stack, declarative memorymap). The selected rule chunks are part of the condition as well as the action side of a procedure. The goalhandler selects the rule chunk that is part of the condition side of the procedure and has a reference to the goal stack. The next step is to retrieve the demand chunk(s) that are linked to the selected rule chunk in order to compare them with the data of the current goal in the goal stack. The first comparison is at the level of the chunktype. The chunktype gives some guarantee that the slot names in the demand chunk(s) equal the slot names in the goal chunk, i.e. the format (DTD) of the demand chunk needs to be identical to the goal chunk in the goal stack. If the chunktype of the goal chunk does not match the chunktype of the demand chunk, the procedure is not selected. Otherwise, the first test—the chunktype comparison—succeeded and a second comparison will take place. This time the contents of the slots in both chunks (demand and goal chunk) are compared. The matching of the contents of the chunk at the slot level depends on the presence of the number of slots to be matched and the presence of variables in the slots and in the slot names.

There are many possibilities to match data. However, in RBot, the mathematical or logical operators that are needed for matching data are kept at the minimum, but enable RBot to do most operations necessary to resemble ACT-R. The following operators or operations are available for RBot:

- *Equal*: slot value of the chunk in the goal stack (or memorymap) equals the slot value of the demand chunk; the goal condition of the procedure.
- *Not equal*: !equal, slot value of the chunk in the goal stack (or memorymap) does not equal the slot value of the demand chunk; the goal condition of the procedure does not match.
- =*null*: the slot value of the chunk in the goal stack (or memorymap) has to be null.
- *Empty*: it is indifferent what is in the slot value of the chunk in the goal stack or memorymap. An empty slot (value) in the demand chunk (condition) of the procedure means that the procedure has no constraints concerning the value in the goal stack or memorymap.

The number of mathematical expressions is kept low on purpose, because they can be handled by a combination of procedures (sub goaling). This is in contrast to ACT-R that uses for example the function eval() and SOAR that applies expressions as for example smaller (<) than or greater (>) than. In the current architecture of RBot, we want to keep the operators at a minimum for simplicity. For specific future releases, an additional mathematical expressions module (eval module) could easily be added to the reasoning engine.

As mentioned in the procedure component description, adding variables to the procedure gives generalisation. The goalhandler builds up a list of possible variables-values combinations (binding) based on the variables specified in the procedure during the condition match. These binding variables and values are used for substitution of variables in other parts of the procedure, e.g. in retrievals or in the action side of the procedure.

Besides that, variables in RBot are used in two places: as placeholders for slot values and as placeholder for variable slot names. The syntax as described in the procedure description before is as follows:

- =variable:
  when the variable is not in the list of collected (binding) variables, the variable is assigned the slot value or the slot name value of the referenced slot of the chunk in the memorymap. Otherwise, the variable is substituted with the already collected variable at the condition side.
- !=variable:
  not equal to a variable, only applied in matching when a variable is already in the list of the collected (binding) variables.

After matching all procedures with the goal chunk, three possibilities can occur:

1. No procedure is suitable to execute: the goalhandler has found no suitable procedure and decides to drop (remove) the goal. If no goal chunk is left in the goal stack, the goalhandler awaits for a new arrival of a goal chunk from the goal feeder.

2. Multiple procedures are selected as possible candidate for execution: the goalhandler, because of serial processing, needs to select only one candidate. Conflict resolution helps in selecting the best procedure based on the highest utility.

3. Only one procedure is selected as a candidate. In this case no conflict resolution is necessary and the selected procedure will be executed in the next phase.

After the selection, the condition side of the selected procedure will be inspected to find out if remaining retrievals have to be made from other memorymaps. If this is the case, the goalhandler will try to retrieve the requested chunk from the referenced memorymap. If the goalhandler cannot retrieve the chunk, because of no match, or the activation of the chunk is too low to be retrieved within time, the goalhandler drops the goal and looks if any other procedure is available in the set of potential procedures. If this is the case, the goalhandler again inspects the new selected procedure to find out if any chunks need to be retrieved. This cycle will go on until all procedures fail or one procedure is selected based on a successful condition side. The next step is the execution of the procedure's action side.

The action side of the goalhandler

The main function of the action side of the goalhandler is managing the goal stack, changing states in other memorymaps of the actor and handling the credit and cost assignment of procedures[20]. In order to change memory states, the action side has three operators: remove(-), add(+) and modify(=). When applied to the goal stack, the goalhandler is able to remove a goal chunk (when a (sub) goal is solved), modify a goal chunk, or add a goal chunk in order to progress and solve a new (sub) goal. The second function of the action side is to change values in memorymaps of the actor. Chunks can be created, removed or changed, thereby changing the representation inside the memory of the actor.

Besides direct action on memorymaps, the goalhandler handles credit and cost assignment of the procedure that is executed. The procedure learns from its past experiences and thereby implements reinforcement learning at the subsymbolic level of the procedure. Next, the goalhandler determines changes in activation of chunks in memory when they are successfully harvested or when chunk merging (adding an already existing chunk) takes place, i.e. the chunk only changes in activation when it is retrieved by a production *and* the same production is executed as well (no retrieval errors). We will now end the discussion of the goalhandler and discuss handlers of peripheral components, such as perception, physical memory and social constructs.

[20]The procedure itself is a bookkeeper of its own experiences.

5.2.2.2 Perception

The goalhandler is occupied with administering the changes inside the actor, and initiates state changes inside the memory. Changes can occur by internal cognition, but also by receiving new events from the environment. The function of the perception memorymap of the actor is to receive and store new events from the environment. The process of receiving events depends on changes of the actor itself, by physical change, or changes that are caused by the environment. The way an actor perceives objects is by receiving signs or signals from these objects, i.e. the semiotic Umwelt (Von Uexküll, 1934, 1957) as described in chapter 3.

Imagine an actor with only visual capabilities in a totally dark room. When light is added to the room, objects become clear because not only the environment is making the objects visible and colourful, but also the actor has the ability to receive the signals (photons or light packages reflected) and signs sent by the object and based on a process of semiosis, the actor interprets the sign and gives meaning to the object.

In the RBot architecture, these types of signs are represented by messages; every object is radiating signs in the environment.

However, it is only the objects or actors that are responding to them that can give meaning to the sign. At this point, the personal physical characteristics of the sensors of the actor come into play. In a dark room, a human being is unable to receive visual signs because its physical design is not capable to receive any information on the retina. However, the mosquito is an insect that has a different physical make-up of the perception. It is sensible to heat convection and smell[21], and is able to detect signals or signs that are not detectable for us. Hence our perception is only able to capture those signs that are based on the physical characteristics and make-up of the actor.

In order to create suitable perception characteristics for the actor, we have to define some physical measures of the distance within which signs can be detected (the horizon). In RBot we have defined a circle with a visual radius surrounding the actor and made the actor only sensitive to a certain segment of this circle; thus simulating a forward-oriented eye. Hence, signs can be detected in a certain area of sensitivity to signs. The second step is to determine for which signs the actor is sensitive. Because signs are represented by messages in XML, the actor is equipped with an XML parser that filters out the messages that give meaning to the actor's personal world. The third step is to transform messages in chunk format so they can be added to the perception memorymap.

The perception memorymap is equipped with a perception handler that sorts messages depending on the predefined physical make-up of the actor. For example, in the case of RBot, signs that are received from objects closer to the actor are more important than signs further away. The handler regulates the importance of the chunks in perception with help of a mechanism that ranks

[21] "Gillett (1971)... emphasizes the importance of temperature in the host-locating and homing behavior of mosquitos. He states that warm moist air currents are the principle cues to distant host location, and that smell becomes important only in the final approach and landing stages of attack behavior". (Davis & Sokolove, 1975, p. 223)

signs or chunks. Because the perception is at the same time defined as a buffer, the signs of less importance are removed from the perception in order to reduce memory load in message overloaded environments.

In order to keep attention to crucial signs, only a certain amount of signs can get attention of the perceiving actor, which is in accordance with the bounded rationality approach (Simon, 1945). The influence of the motion and position of the actor in the 'physical' space on perception makes it clear that the physical properties of movement are connected to the perception of the actor.

5.2.2.3 Action :: physical memorymap and position handler

The physical memorymap has the function to store physical properties of the actor; it is the place where the actor stores its own physical properties in an 'actor' chunk, e.g. its name, network address, its x- and y-coordinate (2D-grid), its visual radius, and so on.

The action or execution side of procedures can steer the motor interface, i.e. they can refer to this memorymap and can change slots that hold physical properties. The simulation experiments, which will be discussed in chapter 6, use procedures that change the x- and y-coordinate in the physical memorymap. Effectors can be attached to the physical memorymap that listen to changes in the physical memorymap.

The position handler is an example of an effector that monitors the current (x,y) position in the physical memorymap and guides the actual movement when these coordinates are changed by sending messages to the server that manages the environment, see figure 5.13. The server notifies other actors that are in the vicinity or range of the moving actor, see section 5.4.2.

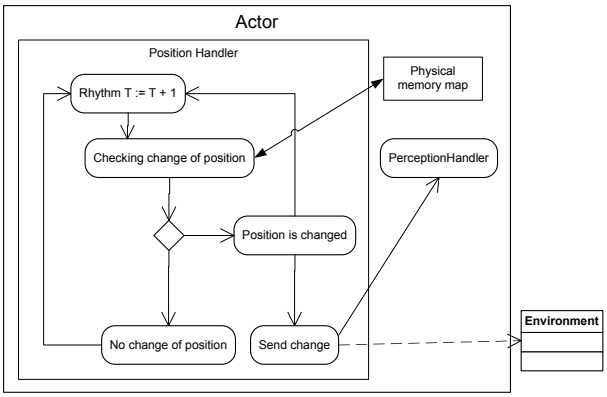


Figure 5.13: Positionhandler.

The change of position needs to be notified to the perception handler as well, because movement of the actor affects the perception of objects as well. The messages that are used for notifying the server are system messages, i.e. they are 'environmental' messages according to a format the server understands in

order to create a change in the presentation of the position of the actor in the task environment.

Besides the system messages, there are other messages that concern the exchange of knowledge or information between actors. These messages are produced by the message handler of the actor that takes care of transforming knowledge or information (e.g. chunks) into messages that are understood by other actors that 'speak' the same language.

5.2.2.4 Action :: message memorymap and message handler

In RBot, we have defined a separate memorymap that buffers outgoing messages that need to be sent to other actors. Attached to the memorymap is a message handler that reacts on so-called message chunks that are stored in the memorymap. When message chunks arrive in the memorymap, the function of the message handler is to convert chunks into messages that can be transported over a network and allows the actor to communicate with other actors, i.e. the moment a message chunk arrives in the memorymap, the message handler has the function to find out what has to be parsed into a message and orders the MessageCreator to create a message out of a formatted chunk in the memorymap.

The formatted chunk or message chunk is created by a 'message procedure', i.e. a procedure that has at its action side an action component that consists of a pattern chunk that can be filled in with properties of the receiver, the knowledge and the name of the object being sent.

For example, in the third simulation experiment in chapter 6 that is concerned with the transfer of a social construct (or goal, or procedure) towards another actor, the message chunk can have the following format/properties:

Chunk name: message_x	ChunkType: Message
Receiver	actor name
Object	socialconstruct | goal | procedure
Objectname	name

Table 5.2: Format of message chunk.

In case of a social construct, the message handler dismantles the message chunk and asks the MessageCreator to create a message, after which the handler sends the created message away to the receiving actor(s) via the server.

The mechanism for receiving messages happens in a similar but reversed way. An actor sends a message to the receiving actor via the server. The message arrives in the message memorymap or buffer of the receiving actor in a language or format that is understood by the receiving actor, i.e. the actor receives messages in its message memorymap or buffer and parses the message/transforms the message into internal chunks or commands understood by the actor's architecture. The communication and formatting of those messages will be discussed in detail in section 5.4.1.4.

5.2.2.5 Memory components and estimating the sub-symbolic properties

Up to this point, the structure of the memory has been explained and the necessary handlers for regulating state changes and receiving input from the perception and sending messages to other actors or the server have been discussed.

However, some issues concerning administering and estimating or calculating the sub-symbolic properties need to be addressed as well. Two software classes, the ActivationMath class and the SubSymbolicMath class are responsible for calculating the activation of a memory component and the sub-symbolic properties of procedures at a certain time.

A memory component keeps track of the activation of a chunk and maintains an array of presentation times of itself. The length of the array can be configured during start up for every actor and depends on the amount of working memory available in the computer and the configuration of the decay factor[22]. An example array with a length of four places is given:

$$[3, 7, 12, \emptyset]$$

When a handler, such as the goalhandler wants to know the activation of a memory component, the memory component calls the ActivationMath class that calculates the activation of the component based on the *base-level learning* equation for activation. When the current time for example equals twenty discrete time steps, the activation of the memory component is:

$$Activation = \ln\left((20-3)^{-0,5} + (20-7)^{-0,5} + (20-12)^{-0,5}\right) = 0,87$$

Associative strengths of ACT-R are also implemented in the ActivationMath class. One of the three methods of calculating activation spread[23] can be applied in RBot at the moment the goalhandler wants to retrieve a chunk from memory.

The other class, the SubSymbolicMath class, takes care of the sub-symbolic properties of the procedure chunk, computes the requested property, for example the expected gain of a procedure, and returns the calculated property back to the component that requested for the computation of a property.

Both these classes that handle sub-symbolic properties have an indirect influence on the performance of the cognitive engine. The social construct handler, discussed in the next section, also has indirect influence on the performance of the cognitive engine, but its mechanism (a meta-cognitive mechanisms) is different compared to the sub-symbolic cognitive mechanisms.

5.2.2.6 Social construct handler

All previous handlers are *directly* dependent on notification of time. Social constructs on the other hand do not only depend on the time aspect, but also respond to changes in the situation inside the memorymap(s) for which it has registered itself. A social construct can register itself for example at the declarative

[22] The larger the decay, the smaller the chosen length of the array. Because, with growing time difference between current time and a presentation, and a large decay, the influence of the presentation gets smaller and can eventually be neglected.

[23] See section 4.4.3.1 in chapter 4.

memory as a listener for events. When something happens inside the declarative memory by which the social construct is triggered, the social construct handler determines if the event needs to result into an action, e.g. in the form of a change in the utility of procedures that are known to be affected by this construct.

Norms are attached properties of the social construct, and they can be explicitly formalised. The *condition* side of the norm can be triggered depending on:

- A chunktype or slot condition of a chunk to which the condition should respond
- The registered memorymap, a map to which the condition side is listening, it only listens to events created in this memorymap.
- A trigger time: the condition side is only triggered when the condition is still present after a certain waiting time.

When the condition side is triggered, the action side is fired. The action side can have impact on many components in the model. For example, the action could affect goal deliberation, other norms, affect the utility of procedures, or generate other actions such as sending messages to certain actors. We refer to chapter 3 that discusses a number of social constructs with attached norms, and their possible effects.

In the current implementation, social constructs are able to change the utility of procedures directly without reasoning so that they have an immediate effect on behaviour, see figure 5.14.

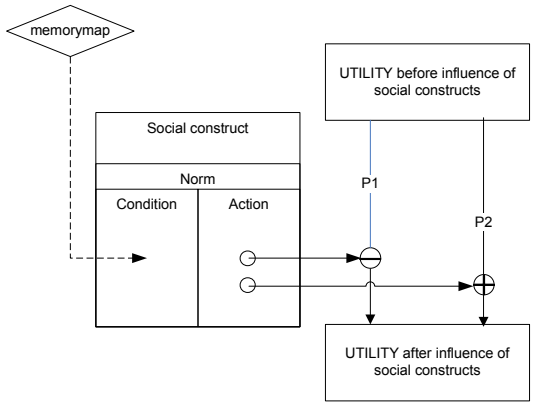


Figure 5.14: Social constructs and the impact on utility of procedures.

For instance, in figure 5.14, we see the influence of the social construct on procedures P1 and P2. When a chunk is arriving in a memorymap, the norm condition is notified and checks whether it can trigger an action. If the condition is true, the norm becomes active and takes action by changing the preferences of P1 and P2. P1 is, for example, demoted and P2 is promoted.

The section about controllers and handlers has shown how the architecture handles events and how this can be controlled. The next section explains the view of the actor RBot that enables the user to have control over the actor.

5.2.3 View and control of the model

The view of the actor RBot is implemented in a user interface (figure 5.15). We will discuss the most important functions that allows the user to (indirectly) control the *stand-alone* RBot actor[24].

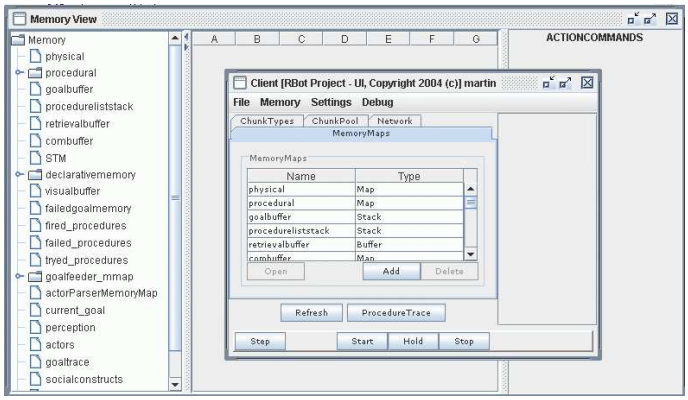


Figure 5.15: User interface of RBot: view and control.

In RBot, as in most simulation environments, the user interface allows the user to control the simulation parameters as follows:

1. *Before running the simulation*: configuring the actor and simulation parameters. Initialising a model can be done by configuring the actor's properties with the help of configuration files; files formatted in XML that can be edited with an XML editor.

   Another important aspect is the configuration of the RBot simulation parameters (based on ACT-R) before launching an actor. Examples of such parameters are the decay rate of the memory components, the noise of the utility function, and so on.

   After launching the actor, the XML configuration files can be selected and loaded into the memory of the actor, i.e. the procedures file, declarative facts-file and the goals for the goal feeder are loaded into the memory of the actor by parsing the files and transforming the files into chunks. After configuring the actor, the simulation can be run.

2. *During the simulation (control and inspection)*: the simulation can be controlled by starting, pausing, and stopping it and making a simulation step

[24]The user-interface of the MRS with more actors will be discussed later on in section 5.4; the mixed tier.

at a time. During the simulation, it is possible to inspect various parts of the actor. For instance, the Memory View, as shown in figure 5.15, allows the user to inspect the contents of the memory—chunks, procedures, social constructs. Another option is to look at the procedure-trace that allows for inspection of the interaction between the goal stack and productions. The debug option allows checking the handlers (e.g. the message handler) with help of traces (messages sent and received) in a debug panel.

3. *Presentation of output (graphs and files)*: the output of the information is stored in files and in database tables. After the simulation is finished, files are generated[25] that can be read with a text-editor. An example of such a file is the file generated by the goalhandler that shows the trace of the goalhandler cycle.

   The data in the database tables represents values, such as utilities and activations of memory components. The presentation of data is handled by a tool whose user interface is integrated as part of the user interface of the mixed tier, see section 5.4.3.

The main purpose of the user interface is to configure RBot in a user friendly and convenient way, inspect the actor's behaviour and get results after the simulation finishes; results that can be used for explaining the behaviour of actors. Some of the output will be stored in files and other output in a database or backend server from which data can be extracted for presentation in tables or graphs.

5.3 Backend server

During every simulation run, data can be stored in files or in a database. This is necessary when simulation data needs to be analysed after the experiment finished. Every actor has a direct connection to the database (see figure 5.16, model 1) to prevent additional overhead in the network; in opposite to the alternative, in which data is first sent to the server and then stored in the database (see figure 5.16, model 2).

Moreover, model 1 allows the actor to run stand-alone without the server and store its data in the database. Secondly, the actor is independent of the server and allows it to function as an actor in a Point-to-Point environment (see section 5.1.2), i.e. it can possibly act as a client as well as a server in a multi-actor environment.

The user has the possibility to select what data of the actor needs to be stored, e.g. activation of memory elements or position and time data. When the simulation starts at time zero, the actor first stores its begin configuration in the database. Subsequently, the actor stores every time step data, *only* when changes in the memory or other events occur; this in order to keep the amount of data being transported at a minimum.

[25] This is also a configuration parameter, which can be set to choose which output files need to be generated.

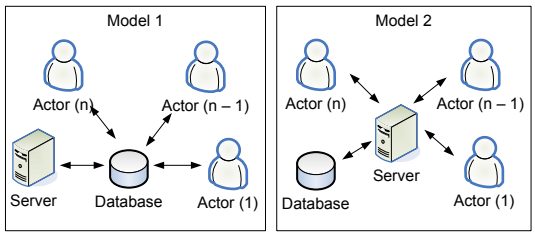


Figure 5.16: Possible database connection models.

In the current implementation, the actors have the function to store and not present their stored data. The mixed-tier is responsible for reading the data from the database and presenting tables or graphs to the user based on the data.

5.4 The Mixed-Tier

The mixed-tier component, i.e. the server, the task environment and the data presentation, has been designed as a central component. The mixed tier is responsible for control and visualisation of the task environment, and presentation of the output of the (inter) individual experiences of the actors and the environment in which they live. Moreover, the advantage of central control is that the complete simulation can be controlled from one control centre instead of controlling every actor separately; even when all actors are dispersed over multiple computers. The mixed-tier fulfils several roles, i.e. it has three main tasks:

Server: control, registration, synchronisation and communication The first task is to control the properties of the environment, registering new actors, and thereby allowing the server to control the synchronisation of processes between the distributed actors. The communication is controlled by the server and regulates (1) the (environmental or system) communication and synchronisation between actors and the task environment, and (2) the communication (ACL[26]) between actors.

Task Environment The mixed-tier also takes care of representing the task environment and visualising and communicating the environment to both, the actors and the users that see the environment being presented on the computer screen.

Graphs and data presentation The third task of the mixed-tier is to act as a client of the backend server. The client creates queries to be executed by the database that returns the correct data for presenting graphs. The data and graphs allow the researcher to interpret the behaviour of individuals and the changes of memory structures of each individual actor.

[26] Actor (or Agent) Communication Language

5.4.1 Server: control, registration, synchronisation and communication

As mentioned before, the server controls and determines the properties of the environment, takes care of the registration, synchronisation of processes and any communication that takes place between actors or actors and task environment. The purpose of central control is to get complete control over all the events that eventually may happen during a simulation run.

5.4.1.1 Control of the simulation task environment

The control of the server is restricted to the environment in which the actors live. The (cognitive) processes internal to the actor are in control of the actor itself, thereby giving the actor as much autonomy as possible.

Most of the control is possible during the initialisation phase of the simulation. Before the simulation starts, the user is allowed to specify the properties of the task environment, e.g. determining the number of actors that can take part in a simulation run. After registration of the actors, the only control the server has is the synchronisation of time and processes between the server, the actors and the task environment. Apart from that, it is possible to start, pause or end the simulation. But before any actor can be controlled, it first has to register itself.

5.4.1.2 Registration of actors

When the simulation starts, the server will first start up the task environment with its environmental parameters, start up the view and controller, and will wait until an actor connects to the server. After starting the server and task environment, an actor has to register itself in order to be represented in the task environment. The registration process is shown in figure 5.17, in which there is first a connection at the level of the network and after the established connection, the actor sends a 'register' request to the server.

The server parses the incoming message and decodes it as a register request. The request contains information regarding the physical properties of the actor that enables the server to create a symbol of the actor. The symbol is added to the symbol attribute map, after which the server is able to create a drawing of the actor in the task environment. The server sends a confirmation back to the actor, causing the actor to be aware of being registered at the server and being able to receive messages from the server concerning the physical environment and sending or receiving messages from other actors.

5.4.1.3 Synchronisation of processes

To control the behaviour of actors that can be distributed over many computers, and in order to replicate experiments in a controlled way, it is important that all actors become aware of the same time and the notion of environmental events and other actors' events.

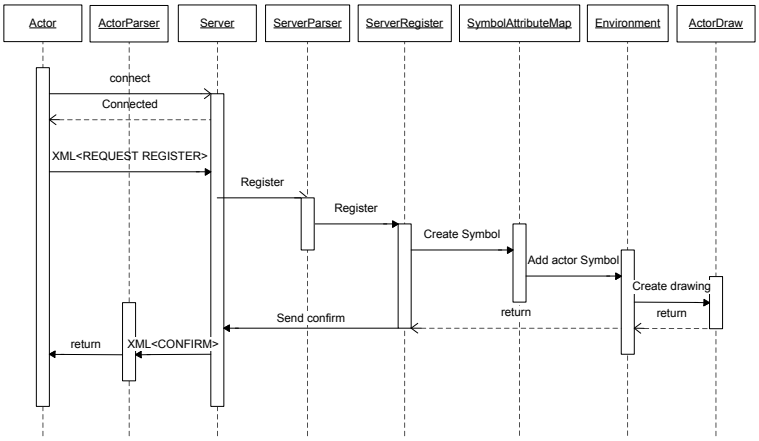


Figure 5.17: Registration of an actor.

Synchronising processes and events can be established with help of a *simulation clock*. If each independent process publishes its current advance in time with help of a simulation clock, then it should be possible to synchronise these parallel processes. In order to decide how to synchronise processes, we first discuss the simulation clock.

There are two principal approaches for advancing the simulation clock: *next-event time advance* (or discrete event simulation) and *fixed-increment time advance* (or discrete time simulation) (Law & Kelton, 2000; Mitrani, 1982). The next-event time advance approach is shown in figure 5.18.

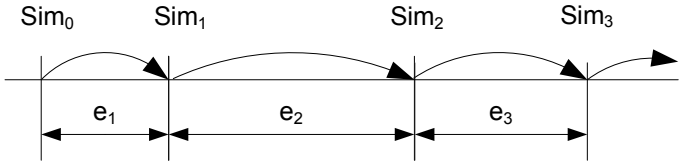


Figure 5.18: The next-event time-advance approach.

> With the next-event time-advance approach, the simulation clock is initialised to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the time of occurrence of the *most imminent (first)* of these future events, at which point the state of the system is updated to account for the fact that an event has occurred, and our knowledge of the times of occurrence of future events is also updated. Then the simulation clock is advanced to the time of the (new) most imminent event, the state of the system is updated, and future event times are determined, etc. (Law & Kelton, 2000, p. 8)

Hence, in figure 5.18, the events (e_1, e_2, e_3) determine the simulation clock advance (Sim_1, Sim_2, Sim_3), i.e. a discrete event simulation makes use of such a simulation clock. On the other hand, with the fixed-increment time advance, see figure 5.19, the simulation clock is not determined by events, i.e.

> ... the simulation clock is advanced in increments of exactly Δt time units for some appropriate choice of Δt. After each update of the clock, a check is made to determine if any events should have occurred during the previous interval of length Δt. If one or more events were scheduled to have occurred during this interval, these events are considered to occur at the end of the interval and the system state... [is] updated accordingly. (Law & Kelton, 2000, p. 9)

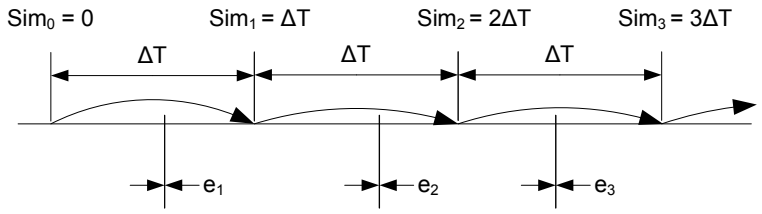


Figure 5.19: The fixed-increment time advance.

With fixed-increment time advance, the simulation clock is independent of the events (e_1, e_2, e_3) that happen during the simulation, i.e. the events are accumulated and calculated at the end of the period.

From both time-advance mechanisms, the next-event time advance is the most applied, because the fixed-increment only is useful for simulating events that are scheduled to be released regularly (controlled), e.g. the expiration date of options in economic simulations every month.

In a Multi-Agent System, we are dealing with a number of different simulators (actors) that need to be synchronised with the task environment and each other. Because actors may run at different computers and may do different tasks, they may easily get out of sync with other actors and the task environment. In case of simulation of multiple processes, we take a next-event time-advance approach in which the synchronisation of the clocks of all registered actors is an event that determines the simulated time, i.e. the simulation clock is a multi-actor synchronisation clock (see figure 5.20).

Figure 5.20 shows two actors, both with a next-event time line. In the first period (ΔT), actor 1 and actor 2 both generate three (not necessarily similar) events (e_1, e_2, e_3). Actor 1 consumes more time than Actor 2 and synchronisation takes place after Actor 1 has finished its events. In the second period ($\Delta T'$), we see that Actor 2 generates an additional event and needs more time than in the previous period. The simulation time in period $\Delta T'$ is determined by actor 2, after which again synchronisation of events takes place. During the simulation, the actors are running in sync, insofar that internal events (e.g. the cognitive

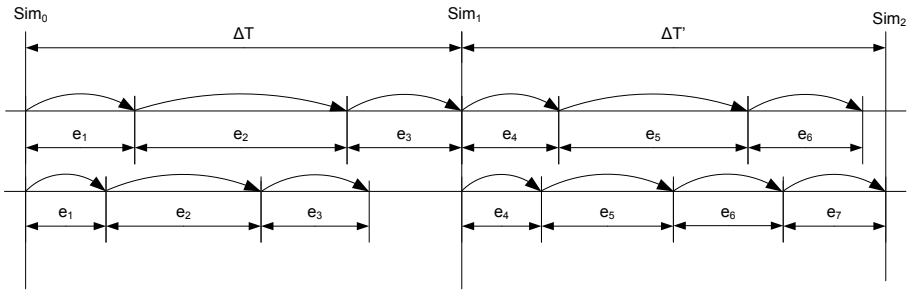


Figure 5.20: Multi-actor synchronisation clock.

engine) are not controlled by the synchronisation clock and are independently executed.

The purpose of the simulation clock is to keep the actors as much as possible in synchrony. Otherwise, one actor could experience a state change (of the environment) a fraction of a time earlier before the other actor would. An actor out of sync with the environment or other actors could perceive the world as it would be a while ago or as it would be a minute from now. Such difference can have influence on what is being perceived by several actors at the same time, i.e. each actor could perceive a different world surrounding him and react in a different way as normally would happen in a synchronised world. Therefore, synchronisation is important, and to keep processes of actors as much as possible synchronised, we have implemented a timer communication mechanism that distributes time to actors based on a time-protocol, see figure 5.21.

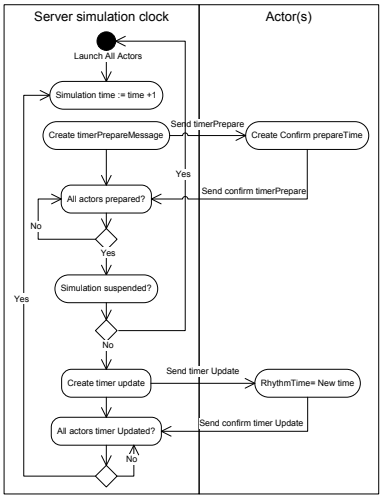


Figure 5.21: Timer communication mechanism.

When actors register themselves at the server, they register at a pool of actors that is monitored by the server. The server tries to keep the actors synchronised

by keeping the times of the actors in this pool synchronised with help of a time protocol. The protocol is a double commit protocol in which the server asks the actors to prepare themselves for a time change. When all actors are ready, a timer update message is sent to the actors and when the actors have sent the confirmation, the process starts over again. Controlling and synchronising the time of actors is one of the difficult aspects in truly independent actors, caused by, amongst others, the following physical conditions:

1. Actors can live on different machines with different processor speeds that can create differences in execution and response times.
2. Small delays in network and serial queueing can easily cause actors running out of sync.

Syncing actors and keeping the awareness of time equal between many actors is a study beyond the aim of this dissertation, but is something that certainly needs to be addressed in studies about distributed real-time systems.

As explained before, in the design of RBot, the actors are running closely in sync with help of a simple double commit time protocol. The communication that is necessary to let the time-protocol function is described in the next section.

5.4.1.4 Communication among actors & between actors and task environment

The main job of the server is to create a task environment—semiotic Umwelt—in which actors can communicate and perceive each other. This allows them to exchange social norms and other events with the help of perceived actions (gestures) and the exchange of an actor language.

As mentioned before, the communication between actors and task environment can be divided into two types of communication.

1. Communication that controls the synchronisation process between actors and the environment; and
2. Communication between actors in a language that is understood by all actors participating in the task environment.

Communication between actors and task environment

For actors to become aware of time and space, they have to interact with the task environment to handle changes that occur in the task environment. The first, as we already have stated, is the awareness of time that helps synchronising the actor with the task environment. The other issue is the physical awareness of space and objects in the task environment. This awareness is controlled by the server that sends (environmental or system) messages towards the actor when another object comes in the visual range of an actor.

Objects in the task environment are changing due to actions of the actors in the environment—e.g. the actor sends messages the moment its location changes—or the environment itself is changing, e.g. weather change. The communication—controlled by the server—between the actors and the task environment will be elaborated in section 5.4.2.

Communication among actors :: an actor communication language

In chapter 3, we have analysed the message-channel as a possible medium for the transmission of someones intentions or knowledge to other actors. We have stated that communication should not only exist out of transmission alone, but also should include the social context and a signification system.

In our MAS, we implement the transmission of signs with help of a standard language (FIPA-XML) that exists out of speech acts. As explained in chapter 3, messages can have the following form, see table 5.3.

```
(performative
  :  sender actor x
  :  receiver actor y
  :  content (...)
  :  language XML
  :  ontology Social Construct DTD
)
```

Table 5.3: General markup of the actor message.

The *performative* (e.g. inform or request)[27] defines the communicative or speech act that states what the sender's intentions are (and what to expect from the other). As explained in the section about the message memorymap, the messages can for example consist of social constructs, goals or procedures. An example of the content of an RBot message that holds a social construct is shown in table 5.4.

```
<socialconstruct name="rightlanedriving">
 <norm name="rightsidelane_norm">
  <normcondition name="rightsidelane_condition" type="Actor">
   <memorymap name="perception" triggertime="0"/>
  </normcondition>
  <normaction name="motivate_evasive_right">
   <affect name="evasive_manoevreRight"
           target="procedure" motivation="10"/>
  </normaction>
 </norm>
</socialconstruct>
```

Table 5.4: Content of RBot message.

The language used for expressing the message is XML and the DTD associated with the message is the 'social construct DTD'. The receiving actor has a signification system that parses the content of the message (e.g. the content shown in table 5.4) based on the social construct DTD and creates a social construct in its memorymap that has the following meaning for the receiving actor.

[27] See http://www.fipa.org/specs/fipa00037 for a complete list of FIPA performatives.

> When a chunk of type 'Actor' is perceived in the perception map, then (triggertime = 0) the norm action is immediately triggered, which states that the target procedure with name 'evasive_manoeuvreRight' should be stimulated with a motivation of 10.

Besides the transfer of signs and the signification system, the task environment (medium) also has impact on the communication between two actors. The communication between actors can be closely linked with the properties of the environment. For example, when an actor wants to say something to another actor, it should produce high enough volume (a physical property of the sending actor) to reach the other actor. Still, whether the other actor can receive the message, depends on the radius in which he still can hear (also a physical property).

In our current version of RBot, the model of speech is defined very simple and a message can always be received by the other actor as soon the actor sends the message towards the other actor. The task environment will be discussed in the next section that elaborates more on how actors perceive each other, which is also based on messaging.

5.4.2 The task environment

The current task environment, see figure 5.22, is created specifically for the experiments described in the next chapter. Hence, a task environment should be redefined for every new and specific simulated environment. For example, the well known *world of blocks* (Minsky, 1985, p. 21) should be modelled and graphically designed for that specific environment[28].

Direct feedback (especially 2D and 3D graphics) about the progress of interaction between actors gives better options for debugging the simulation. Moreover, as observer, it is easier to estimate whether the actors behave according to the design.

The construction of the view works the way a monitor builds up its screen. The environment (or game) panel has a refresh rate and during one refresh the environment is constructed, e.g. the grid, the actors and there properties and the connections between actors. The view will not change as long as the underlying data is not changed.

In the task environment, perceiving others is defined by senses (a semiotic Umwelt). The human being has the following senses: touch, smell, taste, hearing and vision. Vision is the most important perception organ for perceiving the environment (especially when a computer gives visual feedback), therefore we have concentrated our efforts on vision in the current version of RBot. Vision is implemented by creating a visual radius around the actor that determines what objects fall within the perception range of the actor. The perception range is further delimited by the actor's eye angle that is focused towards the direction the actor faces.

[28]Such an explicit graphical design is not always necessary and could be simulated with help of matrices of data points

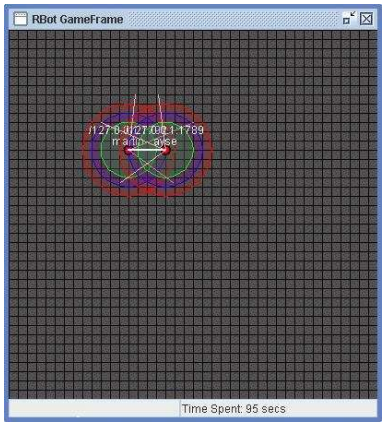


Figure 5.22: A task environment in action.

The stimuli that are sent to the actor are determined by the environment and not by the actor itself. The server determines whether an object or actor is within the perception range of the actor and sends a stimulus in the form of a chunk to the actor that holds information about this perceptible object or actor. The actor filters this information with help of the perception handler in order to react only to messages that are of importance to the actor.

In the case of hearing, the actor has a single centred ear that receives signs that fall within the aural radius. The speech radius determines how far communication can reach, i.e. the "sound" the actor produces has a certain range in which other actors are able to hear it. Hence, the actor has to be in range of the other actor's speech radius to hear what it says. In our architecture, the environment and the distribution of signs (messages) are controlled by a central server that manages the spreading of signs to other objects in the environment, see figure 5.23.

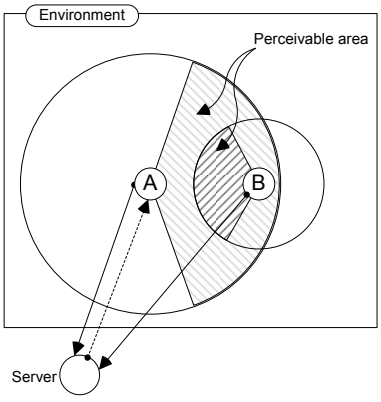


Figure 5.23: Server controls who perceives what.

The environment is constructed with help of messages that carry information necessary to draw actors and objects in the environment. The actor sends messages that concern its initial physical properties, the moment it registers itself at the server. After registration, the actor only sends messages (arrows directing towards the server in figure 5.23) when its physical properties change, e.g. the actor moves from position (x_0,y_0) to (x_1,y_1).

The server on its part only sends messages (the striped line with arrow pointing to actor A in figure 5.23) when they are relevant to the actors or objects.

In case of figure 5.23, the server knows the locations of the actors and therefore is able to calculate the distance between actor A and actor B. Assuming the server has received physical properties from both actor A and actor B, it is able to determine that only actor A requires a message indicating the presence of actor B; actor B is not made aware of actor A. After actor A receives the message of the server, it is filtered by its perception handler that concludes that actor B is perceivable, otherwise it ignores the message.

Another aspect the server takes care of is the process of transforming data describing the physical properties (e.g. position, size...) of the actors in the environment into graphics that can be displayed on the screen.

A relatively simple case is when an actor sends information about its physical properties towards the server. The server sends the data towards the task environment that immediately draws the image to the screen.

A more complex case is the previous one projected in figure 5.23. The following figure (figure 5.24) shows what happens during the moment an actor perceives another actor and the moment the actor moves out of sight of the actor.

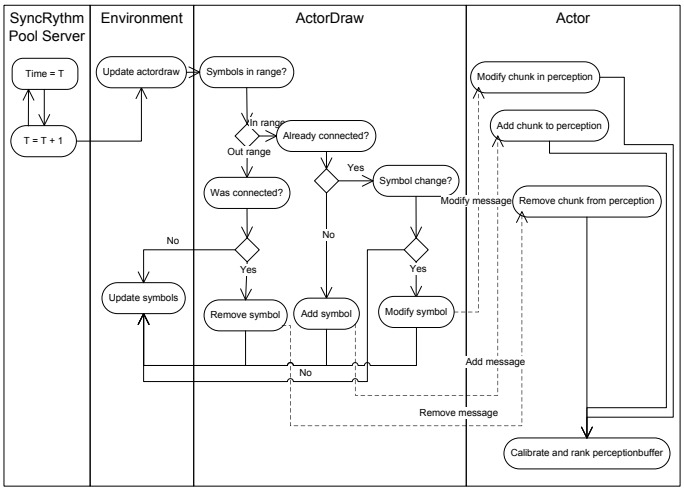


Figure 5.24: Communication between actor and environment.

Every time step, the server requests the environment to update the drawing (sprite) of every actor. The data (physical properties) on which the drawings are based, are stored in the memory of the environment as symbols or attributes.

Every time step, every drawing object wonders if there are other symbols (drawings) that overlap with themselves.

Three situations can happen: (1) the moment the drawing object discovers that another object falls within range and the symbol of that object was not in range before (not already connected), then the symbol is added to the set of symbols that hold all other symbols (drawings) that are in range, (2) the drawing was already part of the set and only the symbol properties have to be modified, and (3) the other object moves out of the range and therefore the associated symbol is removed from the set of connected symbols (drawings).

The previous estimation whether an actor is in range of another actor is handled by the environment. When such an event happens, the environment sends an XML formatted message with the physical properties (e.g. the position) of the other actor (or object) via the server to the relevant actor. The actor that receives the message is equipped with an environmental message parser that transforms the message to an addition, modification or removal of a perception chunk in the perception memorymap.

This scenario describes what the function of the environment is: (1) keeping track of physical properties of objects in the environment, (2) estimating whether actors or objects in the environment collide in the environment, and (3) create drawings of every object in the environment. The number of events that can happen in an environment are many more, such as changes that are independent of actors, e.g. weather, rivers, differences in height.

However, the modelling of a complex environment is beyond the aim of the dissertation. The creation of the task environment only has the purpose for supplying a simple arena that allows simple experiments as described in chapter 6 to be displayed.

The next section will cover the client that is part of the mixed-tier interface, which is concerned with graphs and data presentation.

5.4.3 The client: graphs and data presentation

The last function of the mixed-tier we want to discuss is the data presentation. Based on the results that are extracted from the database in the backend server, the researcher can look back in history and estimate what actually happened during the simulation. The interpretation of data is important, and can be simplified by developing a client with a user interface that queries data from the database and presents the results in graphs or tables to the user.

The structure of the data in the database allows the storage of a set of experiments. Each experiment is assigned a unique experiment identifier. Within the experiment, each actor gets assigned a unique actor id. These identifiers allow the creation of queries that retrieve data from a specific experiment and a specific actor. The following figure (figure 5.25) shows the history of a procedure that was three times successfully applied during its life time.

The current implementation of RBot (part of MRS) allows for storing and presenting data of different levels of the actor; intra-individual data—the subsymbolic data as shown in figure 5.25, individual data of the actor—e.g. its physical properties and its memory structure, and inter-individual data—e.g. data of

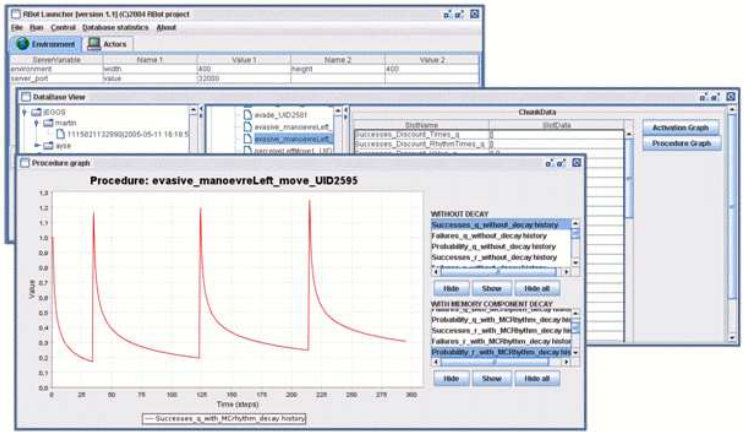


Figure 5.25: User interface that presents data from the database in a graph.

interaction between actors.

A multi-actor simulation creates many events that can be stored in a database or in files. Therefore, in order to prevent overloading a computer system, choices have to be made about what needs to be measured before asking for too much data that unnecessary slows down the simulation.

In the next chapter, we will show results in the form of graphs that are created with help of data that has been collected during a set of experiments. The user interface was the last component of the design that needed to be discussed. In the next section, we will discuss the current design and see if it does fulfil the design requirements we have stated in the beginning of this chapter.

5.5 Discussion

In the discussion we want to reflect and see if our design meets up to the design requirements. Any suggestions to extend or change the architecture, will be postponed to the discussion section or further research in the conclusive chapter 7.

The first requirement states that the architecture should be a Multi-Agent System whose behaviour should be explained in terms of individual autonomous components (actors) and their interaction or relations between each other. Our design fulfils the autonomous aspect by creating individual actors that have their own cognitive engine that makes autonomous decisions based on representations in memory and individual experiences. In a pure classical cognitive architecture, these decisions are made completely independent from other actors. In our architecture, the client server pattern (by creating a semiotic Umwelt as a separate entity) allows for interaction between multiple actors and is an appropriate design for an architecture that can implement interaction mechanisms and delivers a basis for creating social situatedness.

In our opinion, the first requirement has been fulfilled with our design. In

the next chapter, the second experiment will show that actors interact with each other and make autonomous decisions based on their experience with the environment.

The second design requirement demands that a social construct should be a representation in the mind of the actor that can guide social behaviour. In other words, the social construct is a unit of knowledge that allows an actor to respond to certain physical and social situations. A social construct regulates behaviour of the actor at a meta-cognitive or normative level, i.e. the social construct listens to changes in the memory of the actor and gets triggered, the moment a situation appears for which the social construct is applicable. In our design, we have integrated aspects of Brook's subsumption architecture (Brooks, 1986), an architecture that is concerned with the physical environment of the actor and enables the actor to respond with behaviour patterns that match certain situations.

We have extended this idea by not only including physical, but also social situations. The data structure consists of a condition—that listens to changes in the physical or social situation—and an action. The condition-action pair is at the level of meta-cognition, and acts in parallel with the cognitive engine at the lower level. The integration of the subsumption idea allows for an implementation of situated cognition based on ACT-R. Such a modular structure allows for easily adding, modifying, removing and exchanging social constructs with other actors.

The next chapter shows an experiment in which an actor transfers a social construct to another actor. This experiment demonstrates that the subsumption idea applied to an actor is a good choice for implementing coordination mechanisms, such as social norms or contracts between actors.

The last requirement concerns the implementation of a cognitive plausible actor. According to Newell (1980), a physical symbol system is required to hold representations. The ACT-R architecture (Anderson & Lebiere, 1998) is such a physical symbol system and is a (empirically tested) cognitive plausible architecture that fulfils such a requirement.

In our design, we have adopted most of the cognitive mechanisms and structure of ACT-R, but enhanced the design of the memory structure and its components. The addition of meta-cognition and an ACL language parser allows the actor to communicate with other actors and be physical and social situated in the environment, i.e. the actor responds better to influences from the outside world than ACT-R can.

The first experiment of the next chapter shows one of the many possible experiments for comparing ACT-R with RBot. The experiment will not show a detailed comparison, but will show that the behaviour of both architectures is similar.

Although cognitive psychology requires architectures like ACT-R that are precise in simulating the behaviour of the individual, other fields such as sociology and social psychology require architectures that show how individual behaviour is affected by the environment and behaviours of other individuals. In our opinion, RBot (and MRS) allows for studying behaviour of not only individuals (cognitive psychology) but also social behaviour and interactions between actors.

Finally, we want to address the innovations we have applied to ACT-R that has resulted in RBot and MRS.

1. The first innovation is the access and structure of the memory. The implementation of the memorymap, chunks and links in RBot allows for much more complex and flexible structures than ACT-R.

2. Procedures in RBot are small networks of chunks, i.e. chunks are the basic elements in RBot. RBot is therefore more unified than ACT-R that contains separate production structures.

3. The third innovation is the implementation of the variable slot name (DSN chunk). Although ACT-R is discussing the issue, an implementation has not been established yet (version 5.0).

4. The introduction of social constructs in RBot as units of knowledge that are applicable to different situations (physical as well as social) and the integration of aspects of the subsumption architecture is a complete new feature. Discussions about for example meta-cognition are present in the ACT-R community. However, the implementation is not there yet.

5. The formatting in XML is new compared to ACT-R (SOAR already has XML formatting). The formatting of memory structures that allows the memory (components) to be stored and transported over a network is not implemented in ACT-R.

6. Another aspect we want to mention is the creation of a MAS (Multi-RBot System) that consists of a task environment in which more than one actor can live and interact with other actors. Interaction creates new possibilities of not only studying the behaviour of the individual alone, but also the interaction between individuals at the social or organisational level.

7. The final aspect is the introduction of the semiotic Umwelt. The actors are able to perceive the environment and each other by exchanging signs with each other and the environment, i.e. the physical changes are exchanged (communicated) with the environment as well as the exchange of messages (e.g. social constructs, goals and so on) between actors. The semiotic Umwelt creates a (sign-based) environment for actors, and thereby supports physical and social situatedness.

The current chapter explained the design of RBot and MRS with which we tried to fulfil the requirements we have stated in the beginning of the chapter. Right now, the design only has theoretically proven itself. However, such requirements need to be tested and validated with help of demonstrative experiments. This will be done in the next chapter.

CHAPTER 6

Demonstrative experiments with RBot and MRS

THE previous chapter dealt with the implementation of the theory into a working cognitive agent-based social simulation toolkit. In this chapter, we verify and validate the *working* of the toolkit with help of demonstrative experiments, i.e. the experiments will demonstrate that RBot and MRS are capable of modelling social situations with the help of cognitive plausible actors.

In chapter 1, we introduced the design-based approach and simulation as methodology. We explained the life cycle of a simulation study that contains a conceptual model of the real world, the computer model and the experimental model that results in solutions and understanding of the real world situation.

In this chapter, we continue the discussion with regards to the verification and validation of our computer model and our demonstrative experiments, i.e. the chapter discusses how the combination of RBot and MRS—a combined tool for creating simulations—is verified and validated. This will be demonstrated with the help of (1) an experiment that compares the inner working of RBot with that of ACT-R, and (2) two experiments of which one will demonstrate the working of RBot actors in a multi-actor environment (MRS), and the other the working of the social construct as a coordination mechanism for RBot actors that enables them to be socially situated.

6.1 Verification, Validation and Testing

Probably the most important step in developing a model is the stage of verification and validation. Users and developers that use information derived from the results of the model should be concerned with whether the results of such a model reflect the real problem they are trying to research. The concern of getting a successful model is addressed with the help of verification and validation (Moss, Edmonds, & Wallis, 1997, pp. 2–3).

Verification is the process of assessing the goodness of fit to the characteristics of the models empirical referents.

Validation is the process of ensuring that a computer program will not crash and also that it will perform in the manner intended by its designers and implementors.

Testing of a model is ascertaining whether a model is functioning properly, i.e. tests are devised and are conducted to perform either verification or validation or both.

> Some tests are devised to evaluate the behavioral accuracy... of the model, and some tests are intended to judge the accuracy of model transformation from one form into another... Therefore, the entire process is commonly called model *VV&T*. (Balci, 1998, p. 336)

Testing of a model can be done in various ways. For instance, test functions in the model (by creating assertions), control of internal states of objects and the flow of the program in time.

In the field of computer simulation there is an opposite understanding of the terms verification and validation. Although we agree with Scott Moss that verification is in accordance with the correspondence theory of truth and validation is in accordance with the coherence theory of truth[1], there are different views about verification and validation (similar to the debate about coherence and correspondence, see footnote). For instance, researchers in computer simulation and statistics apply the following definition (Balci, 1998, p. 336):

Model verification [internal validity] is substantiating that the model is transformed from one form into another, as intended, with sufficient accuracy. Model verification deals with building the model *right*.

Model validation [external validity] is substantiating that within its domain of applicability, the model behaves with satisfactory accuracy consistent with the study objectives. Model validation deals with building the *right* model.

Nevertheless, we argue that verification/falsification concerns theory or hypotheses, and validity concerns the instrument, in opposite to Balci's verification and validation that refers to the so-called external and internal validity.

[1] A coherence theory of truth states that the truth of any (true) proposition consists in its coherence with some specified set of propositions. The correspondence theory of truth is the view that truth is correspondence to a fact... embracing the idea that truth consists in a relation to reality, i.e., that truth is a relational property involving a characteristic relation (to be specified) to some portion of reality (to be specified).The coherence theory differs from its principal competitor, the correspondence theory of truth, in two essential respects. The competing theories give conflicting accounts of the relation between propositions and their truth conditions... According to one, the relation is coherence, according to the other, it is correspondence. The two theories also give conflicting accounts of truth conditions. According to the coherence theory, the truth conditions of propositions consist in other propositions. The correspondence theory, in contrast, states that the truth conditions of propositions are not (in general) propositions, but rather objective features of the world. (Edwards, 1976)

In computer simulation, however, this distinction is not so clear in comparison to the statistical approach. The distinction in the statistical approach is quite clear, because there is a static number of variables and a clear distinction between dependent and independent variables. We argue that the statistical approach is not applicable to computer simulations, because there are no facilities for handling a complex regulative cycle[2] and it has no facilities for handling feedback loops. Therefore, we apply the definition in accordance with Moss et al.

For the convenience of the reader, we show the figure of the life cycle of a simulation study again.

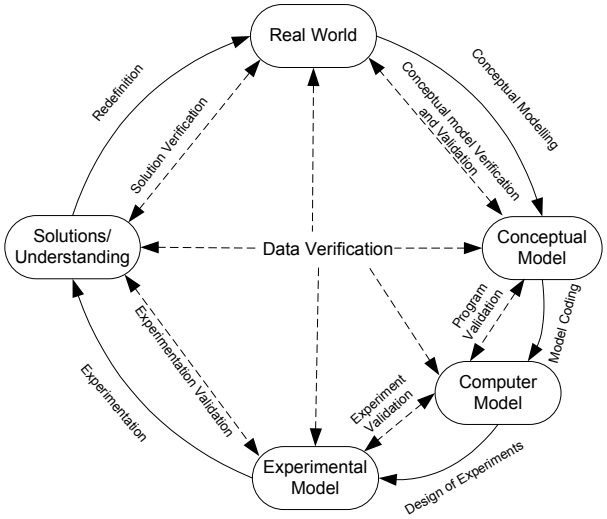


Figure 6.1: Life Cycle of a Simulation Study; see also chapter 1.

Figure 6.1 shows that for each process in the development of a simulation, at least one verification or validation process is performed in parallel (Robinson, 2004, p. 210).

Conceptual Model Verification and Validation: determining that the content, assumptions and simplifications of the proposed model are sufficiently accurate for the purpose at hand and that "the model representation of the problem entity and the model's structure, logic, and mathematical and casual relationships are "reasonable" for the intended purpose of the model" (Sargent, 2000, p. 53).

This can be seen as a comparison of the conceptual model with empirical data in the form of requirements (verification) as well as a check of the consistency of the conceptual model (validation). The question being asked is: does the conceptual model contain all the necessary details to meet the requirements and objectives of the simulation study?

[2]See chapter 1 for a definition of the regulative cycle.

Data Verification: determining that the contextual data, the data required for model realization and verification, and if any data transformations are sufficiently accurate. This applies to all stages in the development process, since data is required at every point.

Program Validation: concerns the accuracy of transforming a problem formulation/converting a model representation into an executable computer program.

Experiment Validation: concerns the accuracy of implementing a specific problem into an experiment model, e.g. initializing the model correctly and creating a model that conforms to a real problem or a predefined artificial world.

Experimentation Validation: determining that the experimental procedures adopted are providing results that are sufficiently accurate for the purpose at hand. Key issues are the requirements for removing initialization bias, run-length, replications and sensitivity analysis to assure the accuracy of the results.

Solution Verification: determining that the results obtained from the model of the proposed solution are sufficiently accurate for the purpose at hand. This is similar to black-box[3] validation (or better verification) in that it entails comparison with the real world.

The most common felt problem with verification and validation is that even when a thoroughly verification and validation process has been undertaken, there are still some difficulties to be recognized (Robinson, 2004, p. 213).

Some difficulties can arise such as (1) there is no real world to compare against (e.g. our artificial society), (2) people have different interpretations of the real world (the perceived world), i.e. if one person claims a model to be verified and validated, someone else claims it is not, (3) the real data collected might be inaccurate due to incorrect collection of data, or a too short time period of collection causing the model to be valid *only* for that specific period and not for other periods, (4) the cost and amount of time needed for validation and verification is often too high to create a sufficient and complete verification and validation process.

> It is often too costly and time consuming to determine that a model is *absolutely* valid over the complete domain of its intended applicability. Instead, tests and evaluations are conducted until sufficient *confidence* is obtained that a model can be considered valid for its intended application. (Sargent, 2000, p. 50)

In the next section, we will explain some methods and techniques commonly used for verification and validation of simulation models.

[3]Black-box validation is applied to determine whether the overall model represents the real world with sufficient accuracy and meets the objectives of the simulation study.
White-box validation is applied to determine if the constituent parts of the computer model represent the corresponding real world; it is a detailed check of the model to make sure that each part of the model does correspond with the objectives of the study (Robinson, 2004, p. 211).

6.1.1 Validation methods and techniques

Probably the most important part of research and of creating simulation models is determining the appropriate validation methods and techniques. For a thoroughly validation and verification of simulation models, there exist many techniques to choose from (Whitner & Balci, 1989; Balci, 1998). In this dissertation, we want to provide a collection of techniques stated by Sargent (2000, pp. 52–53)[4].

VERIFICATION

Animation The model's operational behaviour is displayed graphically as the model moves through time.

Event Verification The "events" of occurrences of the simulation model are compared to those of the real system to determine if they are similar.

Face Verification "Face verification" is asking people knowledgeable about the system whether the model and/or its behaviour are reasonable. This technique can be used in determining if the logic in the conceptual model is correct and if a model's input-output relationships are reasonable.

Historical Data Verification If historical data exist (or if data are collected on a system for building or testing the model), part of the data is used to build the model and the remaining data are used to determine (test) whether the model behaves as the system does.

Operational Graphics Values of various performance measures are shown graphically as the model moves through time, i.e. the dynamic behaviours of performance indicators are visually displayed as the simulation model moves through time.

Predictive Verification The model is used to predict the system behaviour, and then comparisons are made between the system's behaviour and the model's forecast to determine if they are the same.

VALIDATION

Comparison to Other Models: Various results (e.g. outputs) of the simulation model being validated are compared to results of other (valid) models.

Internal Validity: Several replications (runs) or a stochastic model are made to determine the amount of (internal) stochastic variability in the model. A high amount of variability (lack of consistency) may cause the model's results to be questionable and,

[4]Some of these techniques can be categorised under both verification and validation

if typical of the problem entity, may question the appropriateness of the policy or system being investigated.

Parameter Variability-Sensitivity Analysis: The technique consists of changing the values of the input and internal parameters of a model to determine the effect upon the model's behaviour and its output. Those parameters that are sensitive, i.e. cause significant changes in the model's behaviour or output, should be made sufficiently accurate prior to use of the model.

Traces: The behaviours of different types of specific entities are traced (followed) through the model to determine whether the model's logic is correct and whether the necessary accuracy is obtained.

The verification and validation techniques discussed here are applicable to *operational validity*. "Operational validity is concerned with determining that the model's output behavior has the accuracy required for the model's intended purpose over the domain of its intended applicability" (Sargent, 2000, p. 54). The validation techniques can be applied either subjectively—e.g. looking at the models behaviour, or objectively—using statistics to get an (more) independent judgement of the model. The other attribute is whether the system is observable, i.e. is it possible to collect data about the operational behaviour of the real world problem (system).

Table 6.1 shows a classification of operational validity.

	Observable system (Verification)	Non-Observable system (Validation)
Subjective Approach	(1) Comparison using graphical displays Explore model behavior and compare with reality	(2) Explore model behavior Comparison to other models
Objective Approach	(3) Comparison using statistical tests and procedures	(4) Comparison to other models using statistical tests and procedures

Table 6.1: Operational Validity Classification[5] (adapted: Sargent, 2000).

Before going to the experiments section, we want to show the combination of two different developmental cycles that are applied in our research, i.e. the single and multi-actor development cycle.

[5] "Comparison" means comparing/testing the model and system input-output behaviors, and "explore model behavior" means to examine the output behavior of the model using appropriate validation techniques (Sargent, 2000, p. 54).

6.1.2 The combined cycles of RBot and MRS

The development of RBot and MRS consists of a combination of the single actor development cycle and the multi-actor development cycle, see figure 6.2.

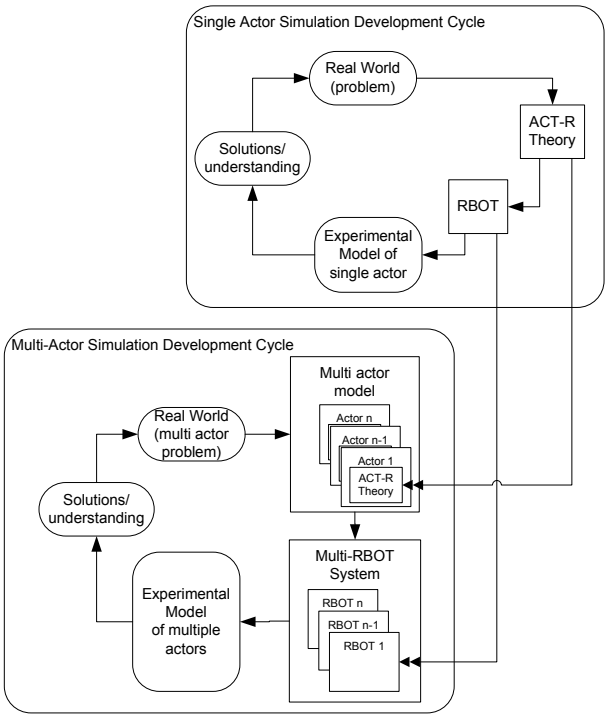


Figure 6.2: The two development cycles of RBot and MRS.

The actor is a separate autonomous component in the model MRS, therefore RBot is developed in a separate development cycle and later on included into the development cycle of the multi-actor simulation. In the next section, we will describe two types of experiments that correspond to the development cycles in figure 6.2; one single actor experiment and two multi-actor experiments.

6.2 Demonstrative experiments

The first demonstrative experiment, as explained before, consists of a single actor experiment. The purpose of the experiment is to compare the behaviour of the RBot model with that of the ACT-R architecture. The experiment is concerned with the validity of RBot and aims at finding out whether the behaviour of RBot matches the behaviour of ACT-R as a cognitive architecture.

The other two experiments describe an interaction between two actors and are concerned with the construction of an 'artificial society'. The implementation of such a society with help of RBot and MRS allows us to test and demonstrate whether certain assumptions about behaviour, e.g. the influence and ex-

change of social constructs, can be made explicit with help of a simulation model. Hence, it allows us to (1) demonstrate that something is possible and (2) suggest ideas (Holland, 1998). We will first discuss the single actor experiment.

6.2.1 Experiment 1: RBot vs. ACT-R

The goal of developing RBot in the first place is to create a multi-actor simulation toolkit with *cognitive plausible actors*, i.e. the purpose of RBot is to deliver a cognitive actor for the simulation and explanation of organisational and social behaviour. RBot's architecture is supported and grounded with help of cognitive psychology and the ACT-R architecture. RBot is part of a task environment (MRS). Therefore, the focus is not alone on experiments with a single actor, but also on the interaction between cognitive plausible actors. The contribution of RBot is not the delivery of a complete new cognitive architecture, because ACT-R already fulfils the role of studying cognitive mechanisms and rational behaviour of the individual. The contribution of RBot, as part of MRS, is to deliver a new actor that, besides its cognitive plausibility, is able to interact (exchange (social) signs) with other actors and serve as a plug-in for other architectures and task environments. The RBot vs. ACT-R experiment is meant as a comparison and not as an exact working copy of ACT-R, i.e. the behaviour of the models is compared. The experiment attempts to answer the following question, as posed in chapter 1:

2.1 Is RBot comparable to ACT-R as a valid cognitive plausible actor?

To answer the question, we have to take into consideration what methods of operational validity can be applied. ACT-R has been empirically verified by comparison with an observable system[6] (the mind of a human being), whereas RBot is validated by comparison with ACT-R[7]. The experiment chosen for comparing RBot with ACT-R is the 'addition-problem'.

6.2.1.1 The addition-problem

The addition-problem of ACT-R (see Anderson & Lebiere, 1998, pp. 30–35) will be used as the control experiment in order to validate (so far as possible) if RBot works according to the principles of ACT-R.

The following validation techniques have been applied: *event validity*—control if certain events that happen in ACT-R also happen in RBot, *internal validity*—many functions were tested and countless reruns have been made, *operational graphics*—with help of graphs, the behaviour of functions and sub-symbolic components is checked and *traces*—with help of data stored in files, processes of functions and sub-modules can be traced. The debugging processes are necessary for *verification and validation* of the conceptual model of RBot. The

[6] Cell (3) in table 6.1.
[7] Cell (4) in table 6.1.

purpose of the experiment is to check operational validity and compare the outcome of the experiment of a single actor with the behaviour of ACT-R (black box validation).

The (multi-column) addition-problem experiment addresses some important aspects of ACT-R that can easily be shown with traces of output in RBot. A complete validation of RBot will not be given in this dissertation, because (1) it is beyond the aim of the dissertation and (2) the time necessary for validation would be beyond the budget of the project. A more complete validation with ACT-R should be done when RBot is applied in purely cognitive experiments.

With help of the experiment, we want to address some of the elements that are crucial for the working of RBot as well as for ACT-R. The addition-problem is an appropriate and easy to understand experiment that simplifies the explanation and validation of many of the basic elements, given in the following list of validation tests.

Test I: Goal stack and procedure flow ACT-R (and RBot) has a goal stack that keeps track of the current goal. The procedure flow, defined by the syntax of the procedures, depends on the current goal, the chunks in declarative memory and the activations of these chunks.

Test II: The base-level learning equation Another point is to check whether the base-level learning equation is present in the chunks of RBot (by validating the activation of chunks over time).

Test III: Retrieval of chunks When in ACT-R a chunk (e.g. from declarative memory) is retrieved, a production is executed, and the activation of that chunk will increase. The increase of the activation value of a chunk should be present in RBot as well.

Test IV: Merging of chunks ACT-R defines that when chunks refer to the same fact, they are merged together and the activation is increased. In our experimental setup, the addition-problem is solved two times in order to show how merging of new chunks works.

Based on the findings, we can determine whether declarative symbolic learning (by adding new chunks) and sub-symbolic learning (by increasing the activation of chunks) are implemented correctly.

General working of the addition-problem and the experimental setup. In order to understand the outcome of the problem better, first a small description of the addition problem will be given.

1. The assumption of the experiment is that for solving an addition problem, the actor will first look in memory to find out whether the answer is already known.

2. If the answer is known, it will be stored in a communication buffer (in order to be picked up by a message handler).

3. If the answer is unknown, the actor has to start solving the addition-problem as a new problem.

In the setup of the experiment, the addition problem is run two times, i.e. the actor is asked to solve the same addition problem two times. In the first run, the actor doesn't know the answer and starts the addition problem. In the second run, the actor is assumed to know the answer and gives the right answer immediately. The experimental setup defines the following addition problem that consists of the addition of two numbers, 5155 and 7777, see figure 6.3.

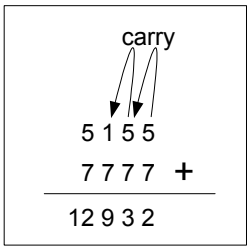


Figure 6.3: A multi-column addition problem.

The assumption is that the actor has no knowledge about this specific problem. Therefore, it is necessary to divide the problem into sub-problems of adding the numbers in one column. After solving the addition of a column, there exists the possibility that the sum of two numbers is larger than nine. In that case, a carry has to be transported (in this problem 2 times) to the next column and added to the result of the next column. In case the single column addition is smaller than ten, no carry transfer is necessary. The complete problem, can be divided into the following sub-problems (simplified):

1. Start at the first column.
2. Add numbers in column + (if necessary) from the right column the remembered carry and write sum.
3. if answer > 9 , remember carry
4. column = column + 1 (go to the next column)
5. go to sub-problem 2 if column < 5
6. if last column done, give the answer

The rules or procedures of RBot and ACT-R are complicated, because procedures for storing sub-problems in certain memory locations are necessary to remember the temporary answer[8]. The complete set of productions of the experiment is shown in appendix A[9].

[8]For example, humans use paper as a temporary memory, so references to temporary storage have to be included in the procedures.

[9]For a comparison, see (Anderson & Lebiere, 1998, p. 7).

6.2.1.2 Results of the addition-problem experiment

We will address the results following the list with tests given in the previous section. The first test we want to elaborate is the goal stack and the procedure flow.

Test I: goal stack and procedure flow

The goal stack keeps track of the current goal and in the addition problem there is a 2 layer goal structure that separates problem (addition-problem) from the sub-problem (add_column). The following table shows the trace of the goal stack[10] of RBot in which the indentation reflects the depth of the goal stack.

Table 6.2: Goal stack addition problem.

Cycle	Procedure	Description	Result
Cycle 0	P1	Start Problem push goal check declarative	Success
⟶Cycle 1	P2	Check addition problem in memory	Failure
Cycle 2⟵	P4	After check declarative memory start problem	Success
Cycle 3	P5	Addition read goal	Success
⟶Cycle 4	P6	Add numbers no carry Retrieve 5 + 7 = 12	Success
Cycle 5	P10	Add column last column	Failure
Cycle 6	P8	Add column greater than 9 Retrieve 10 + 2 = 12	Success
Cycle 7⟵	P11	Addition problem write column abstract Retrieve column + 1 ⇒ 1+1 = 2	Success
⟶Cycle 8	P7	Add numbers process carry Retrieve 1 + 5 = 6 Retrieve 6 + 7 = 13	Success
Cycle 9	P8	Add column greater than 9 Retrieve 10 + 3 = 13	Success
Cycle 10⟵	P11	Addition problem write column abstract Retrieve column + 1 ⇒ 2 + 1 = 3	Success
⟶Cycle 11	P7	Add numbers process carry Retrieve 1 + 1 = 2 Retrieve 2 + 7 = 9	Success
Cycle 12	P8	Add column greater than 9	Failure
Cycle 13	P9	Add column smaller than 10 Retrieve 0 + 9 = 9	Success
Cycle 14⟵	P11:	Addition problem write column abstract Retrieve column + 1 ⇒ 3 + 1 = 4	Success
⟶Cycle 15	P6	Add numbers no carry Retrieve 5 + 7 = 12	Success
Cycle 16	P9	Add column smaller than 10	Failure
Cycle 17	P8	Add column greater than 9	Failure
Cycle 18	P10	Add column last column	Success

Continued on next page

[10] A similar trace of ACT-R can be found in The Atomic Components of Thought (Anderson & Lebiere, 1998, p. 9).

Cycle	Procedure	Description	Result
Cycle 19⟵	P12	Addition problem write last column	Success
Cycle 20	P13	Addition problem communicate	Success
Cycle 21	P1	Start Problem push goal check declarative	Success
⟶Cycle 22	P2	Check addition problem in memory Retrieve 5155 + 7777 = 12932 Merge with last chunk	Success
Cycle 23⟵	P3	Found and clean	Success

The procedures P1...P23 show successes and failures. The procedures competing with each other are P8, P9 & P10. The three procedures compete because they respond to the same condition in the goal stack. Because P10 ('add column last column') experiences a failure at cycle 4, it will take this experience with it through time. In the competition, from cycle 16...18, it shows that P10 was selected as the last option, because of its failure history in cycle 4. With this trace of procedures in the goal stack, a check has been done on the fact that not only the flow of procedures and the goal stack functions accurate, but also that procedures that experience more failures are less likely to be selected later on because of their lower success ratio. This is valid and accurate behaviour compared to the behaviour of ACT-R, therefore we can with confidence state that the goal stack and procedure flow of RBot behave according to the requirements stated by ACT-R.

Test II and III: The activation of chunks over time and retrieval of chunks

Test II and test III are concerned with testing whether the functionality of activation over time and the retrieval of a chunk in RBot are functioning according to the principles of ACT-R. Test II requires the chunk to decrease in activation according to the activation equation 4.1 (base-level learning), see chapter 4. Test III requires that when a chunk is retrieved, the activation should increase.

In order to see whether those changes of activation of chunks and the retrieval of chunks have a similar effect as in ACT-R, we make use of comparing the base-level learning equation of ACT-R with the graphical output of RBot. Referring to table 6.2 and figure 6.4, we see that both, fact 1+1 = 2 and fact 5+7 = 12, are retrieved two times from declarative memory.

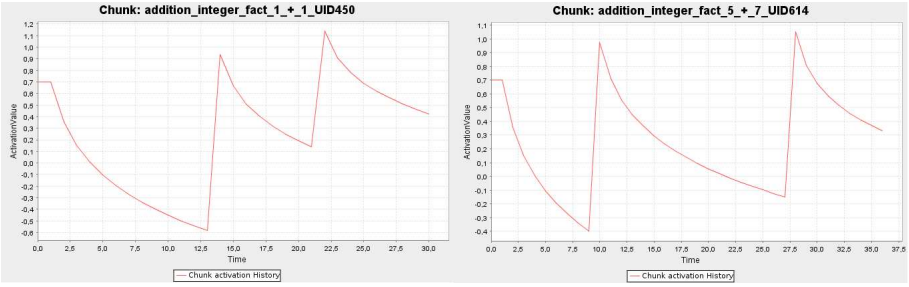


Figure 6.4: Activation: base level learning and retrievals of chunk 1+1 = 2 and chunk 5+ 7 = 12.

Referring to table 6.2, we can conclude, in case of chunk 1+1 = 2, as follows. One time the retrieval was necessary for changing the pointer from column 1 to column 2 and the other time for adding the carry to digit 1 of 5155 in the third column. The other chunk, fact 5+7 = 12, was retrieved for (1) calculating the first column, and (2) calculating the last column. The output of the graphs shows that RBot passes test II and III, and succeeds in implementing the activation function as well as the change by retrieval of chunks; its behaviour resembles the (sub-symbolic) behaviour of chunks in ACT-R.

Test IV: Merging of chunks

One particular aspect of ACT-R is the process of merging chunks (Anderson & Lebiere, 1998, p. 125). ACT-R states that a chunk is an unique object in memory. Therefore, when a chunk is recreated, it is merged with the original chunk already present in memory.

At the end of solving an addition-problem in the experiment, a new chunk (5155+7777=12932) is created and is written to a 'communication-buffer'. In our experiment setup, the addition-problem is solved two times. Therefore, the answer-chunk of the second (sub) problem (also 5155+7177=12932) is merged with the previously created chunk that already exists in the communication-buffer, see figure 6.5.

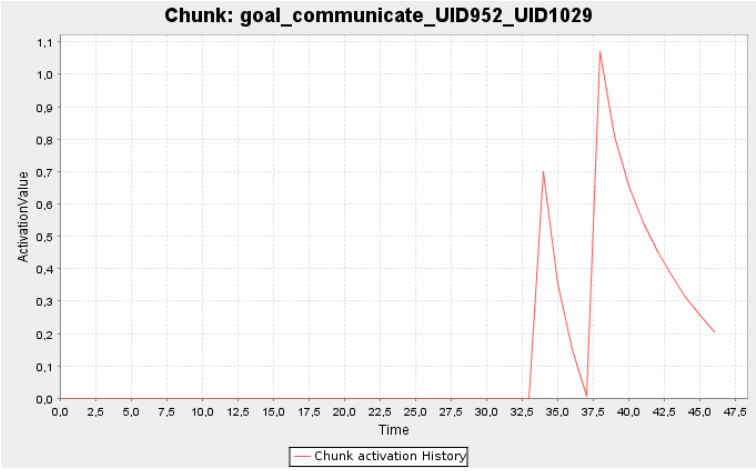


Figure 6.5: Creation of a chunk and the merger of a chunk.

The merging process takes place by adding a presentation time to the already existing chunk that exists in memory. Figure 6.5 shows that the same chunk receives two presentations; the first is an initial activation at time step 33, and the second occurs when RBot attempts to put a chunk in the communication buffer while an identical chunk is already present in the buffer.

6.2.1.3 Discussion

The purpose of this experiment is to compare RBot and ACT-R in order to see whether the basic aspects of ACT-R, the goal stack, procedure flow and the base-level learning equation are working in RBot the way they are supposed to work in ACT-R. Concerning the aspects addressed in the tests, we can conclude that a single actor of RBot behaves in a similar way as ACT-R[11] does.

Apart from showing the similarities of ACT-R with RBot, the first experiment also serves the purpose of getting confidence with the RBot model, which is probably the most important reason for creating test cases and experiments.

We are aware of the fact that in order to create a more complete validation of RBot, many experiments of ACT-R have to be replicated and many other validation tests have to be conducted. However, as mentioned before, such an analysis is beyond the scope of this dissertation and besides that, it would become too costly and time consuming to determine that RBot is '*absolutely*' valid over the complete domain of ACT-R its intended applicability[12]. Nevertheless, the cognitive actor RBot is capable of holding representations in the form of social constructs and operates on these representations in a similar way as ACT-R prescribes. Before going to explain the experiment that implements social constructs as prior knowledge, we will first address the interaction between RBot actors in the task environment (MRS) that discusses the emergence of social constructs caused by interactive learning.

6.2.2 Experiment 2: Emergence of Social Constructs and Organisational Behaviour

In this experiment, we want to address the question whether actor interaction can lead to the formation of social constructs (the interaction aspect of research questions 1.1 and 1.2 stated in chapter 1). More specifically, we refer back to the implementation question 2.2:

> **2.2 Is it possible that social constructs and organisational behaviour can emerge from (social) interaction?**

In order to model such an experiment, we require a model of a task environment and simulation model consisting of multiple RBot actors; our Multi-RBot System (MRS).

The purpose of the experiment is to show the additional extensions of the model by creating a multi-actor system with space and time awareness for the actors and the ability to perceive each other. Hence, we want to demonstrate and find out whether RBot actors can 'live' in a task environment and learn from interaction by cooperation, observation or other means. We have modelled a multi-actor environment of two actors in order to study the emergent behaviour of interaction between two individuals. The interaction between two

[11] The experiment concerns ACT-R version 4.0. ACT-R version 5.0 and beyond dropped the concept of a goal stack.

[12] Of course, we have conducted many tests, but in this dissertation our main goal is to implement social constructs that enable RBot to be 'connected' with the social environment.

actors gives enough material to study phenomena and theories (e.g. management and organisation, cognitive science, AI, to name a few) as previously explained in this dissertation.

Cognitive modelling of organisational behaviour requires that organisations are mental representations within an individual's mind, as explained in chapter 3. For analysis of organisational behaviour, we describe phenomena mainly at the individual level of description as a social construct (the behaviour of the individual actor), but we also describe behaviour at the social level by observing the interaction patterns between actors that emerge during the experiment. Hence, this experiment will analyse behaviour of the individual and its internal cognitive properties and the emergence of behaviour of the collective as well (the behaviour caused by interacting or communicating individuals).

6.2.2.1 Description of the experiment

In chapter 3, we have introduced the social construct as a 'regulator' of social behaviour of actors. In this experiment we will explain how an (internalised) social construct with its attached norms (s) and rules emerges and evolves over time. The experiment will highlight that the actors seem to have a 'social agreement', but they actually adapt their own behaviour to the environment including the other actor, i.e. they do not intentionally influence or change each other's behaviour to come to an agreement.

The social construct and its attached norms have the ability to emerge and create regular stable behaviour in an otherwise chaotic system. Epstein (2001, p. 9) ascribes two features to norms: "They are self-enforcing behavioural regularities (Lewis, 1969; Axelrod, 1986; Young, 1993, 1995). But second, once entrenched, we conform *without thinking about it*. Indeed, this is one reason why social norms are useful; they obviate the need for a lot of individual computing." What Epstein means is that the system creates stability by interaction in an environment that doesn't change much over time, i.e. all actors have conformed to a stable type of behaviour that needs no further attention because of its stability and reduces thereby the cognitive load of the actor. Epstein describes his multi-agent model at a rational level in which entities are described in an economical way.

Our experiment, described in this section, is based on actors that have each a cognitive architecture; an implementation of RBot. The purpose of the experiment is to show that a social construct can emerge not only by looking at the actors' collective outcome in the way they socially behave, but also what happens at the cognitive level of each actor.

Before we start to describe our model of the experiment (see section 6.2.2.2), we first analyse the given problem using game theory/the repeated prisoner's dilemma[13]. This theory models norms or social constructs as equilibrium strategy choice. Experiment 2 consists of a two-dimensional environment in which actors can move freely. The goal of the actor is to move from one side of the field to the other side of the field without colliding with the other actor, i.e. the

[13]See chapter 2 in this dissertation or Axelrod (1984).

actors have to pass one another from the right or left side in order to prevent collisions. Figure 6.6 describes the choices in the environment; a coordination game (see Picker, 1997) in which two Nash equilibriums are present.

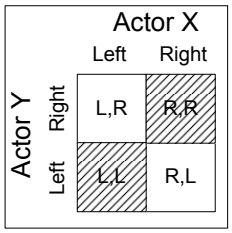


Figure 6.6: Norm selection as a coordination game (Picker, 1997).

Payoffs are the highest when both actors pass right or left. Hence, when selecting the same side to pass, either right or left, then a Nash equilibrium is the outcome. We can extend the model with individual experiences and memory of past experiences and use a simple form of learning, partially based on the utility function of ACT-R[14]. We define the following formula:

$$Utility = \frac{1+S}{1+S+F}$$

Every actor remembers its score and adapts its strategy in the field over time, see figure 6.7.

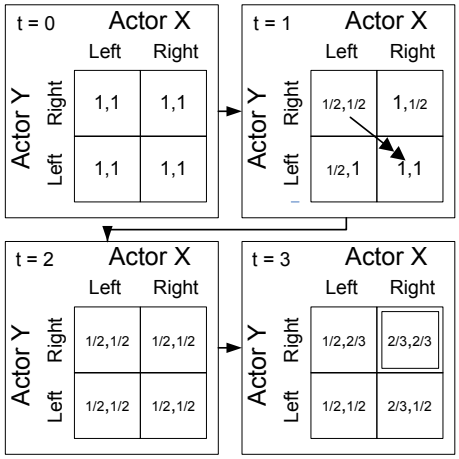


Figure 6.7: Nash equilibrium development over time.

At t = 0, the actor has no preferences and selects randomly a strategy, i.e. in this case actor X left and actor Y right. Both actors receive a penalty of F = 1

[14]In ACT-R, the utility calculation is much more complex. For instance, noise is added to account for some variance in decision making.

and the utility of the selected strategy changes to ½. Now, at t = 1, the opposite strategy is attractive because of its higher utility, i.e. actor X selects right and actor Y left. Another penalty creates a field in which both strategies are valued again as equal. Suppose that both, actor X and actor Y, decide to pass right. Now, both are credited with a success and a Nash equilibrium has been established.

In the above pattern of behaviour, actor X and Y learn over time that one of the strategies, passing both from the left, or from the right provides the highest payoff. The application of game theory to learning behaviour clarifies the pattern of behaviour of the actors for the simulation.

6.2.2.2 The model of the experiment

As explained in the previous section, the simulation experiment (see also Helmhout et al. (2005a)) describes a two-dimensional environment in which actors can move freely. The goal of the actor is to move from one side of the field to the other side of the field without colliding with the other actor, i.e. the actors have to pass one another from the right or left side in order to prevent collisions. In countries like England and the Netherlands, these rules have evolved differently and have been institutionalised by the enforcement of law: passing from the left or passing from the right side, respectively.

In our simulation, we want to study how this norm evolves over time and if a preference for either passing left or right is established between two actors. We illustrate how one of these norms (as part of a social construct) becomes preferred compared to the other by the process of interactive cognitive learning. In the simulation experiment, the actors are given the same parameters, procedures, and declarative chunks, i.e. the initialisation parameters of the actors are equal: equal decay rates, equal utility-preferences, equal motivation values to solve goals, equal procedural and declarative memories, and equal noise distribution functions. The actors have a similar simulation setup and are identical to make sure that the simulation outcome is based on interaction and not on differences at the cognitive level of the actor[15].

The simulation includes two actors, each at the same X,Y grid; one travelling from coordinates 100,100 to 200,100 and back and the other way around for the other actor. In that case, the actors meet up with each other, and decide to pass left or right. In figure 6.8, a screen shot shows that one actor decides to go right and the other decides to go left, i.e. a collision occurs. Based on their perception, the actors notice whether their evasive behaviour was successful or not.

The actor learns in an indirect way; it evades right or left and then sees if its evasion was successful or not. Based on this experience, the actor updates its productions and uses this experience in the next encounter. Figure 6.9 shows the possible states and state-changes (arrows) that are possible for the actor when it executes a production in the experiment.

[15]For example, if we initialise actor X with a higher decay rate compared to Y, it will forget its past quicker than Y and probably will behave differently. However, in this experiment we are interested in the interaction and not the difference between behaviour of individual actors caused by a difference in cognitive parameters.

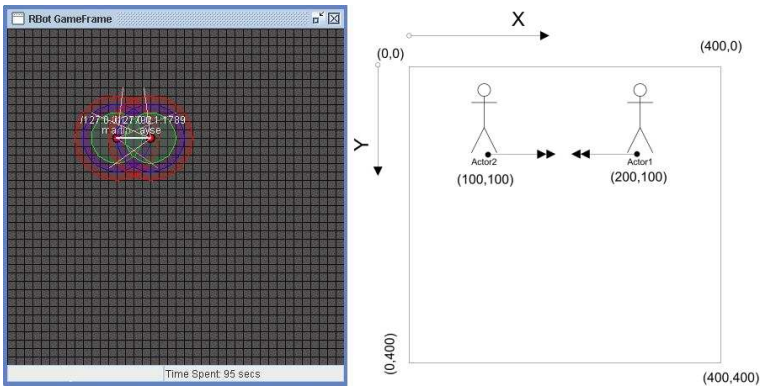


Figure 6.8: Game Frame of two actors trying to pass each other.

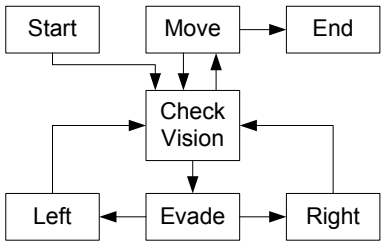


Figure 6.9: States and state changes of the productions of an actor.

After 'Start' the actor checks its vision to see if another actor is present in its range. If there is an actor, it starts to 'Evade', chooses either 'Right' or 'Left' and after evading starts checking whether it was successful or not. If it was successful, the actor chooses 'Move' otherwise it starts evading until it escapes out of the situation by a successful escape manoeuvre. The actor moves until the goal ('End') is reached.

6.2.2.3 Results

According to the game theory, both actors should end up in an equilibrium. When both actors choose the same strategy, they pass successfully and start to reinforce that particular strategy and leave no opportunity for the other strategy to emerge. In the simulation, we encountered the same behaviour when actors chose the successful strategy, but we found an initially unstable behaviour when actors began with a non-cooperative move. Apparently, the experience of immediately success gives the actors the intention to choose the same strategy again, reinforcing successful behaviour and immediately get a lock-in. On the other hand, when selecting initially the wrong strategy, the actors need time to find out what strategy is successful.

In our experiment setup, we are especially interested in the case where both actors initially choose the wrong strategy and have to decide based on expe-

rience which strategy is acceptable for both actors. By initially selecting the colliding strategy, the actors show a shifting behaviour from left to right and from right to left[16]. After some collisions, the Boltzmann factor (noise) in the utility function gives the actor freedom of escaping from the hopping behaviour. This gives actors the opportunity to settle into the same successful strategy, both passing either left or right.

In order to understand the behaviour of the actors, we want to look for behavioural patterns at two levels, one for the behaviour of the actor that is reflected by its internal history of memory and the other in the role of the external observer that spots the actors having a collision and ending up in an emerged pattern. With help of those patterns, the behaviour of escaping and reinforcement can be explained.

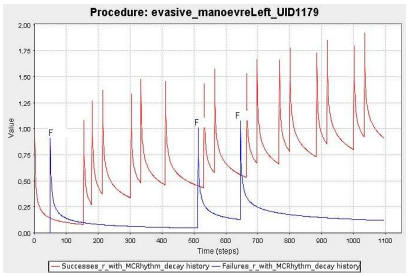
The results that matter for finding out the emergence of a social construct with a preference for passing left or right, are the 'Left' and the 'Right' procedures of the individual actors.

Figure 6.10—(a) till (h)—show the results of the experiment that started with a collision. First, no particular strategy is chosen; the right move as well as the left move is equally preferred. However, after approximately time step 700, the actors prefer the 'Left' strategy based on interaction. Due to this interaction, they develop a preference in favour of the left move so strongly that the actors only choose the left strategy, i.e. the utility difference between right and left becomes so significantly large, that the actors are locked-in into the left passing strategy.

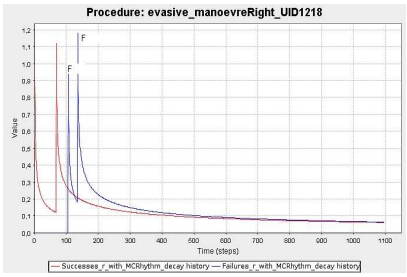
The successes, failures and utility of actor X and Y—figure 6.10 (a), (c), (e) and (g)—indicate that many interactions reinforce the left strategy. These reinforcements cause the actors to behave in the same fashion all the time, under the condition of course that the environment stays constant.

According to the behaviour of the simulation; if we do not look inside the actors, we could conclude to see some start of "formalised" behaviour, in the sense that actors seem to have a mutual agreement. An observer (third party) who looks at the interaction between the actors, can see a clear pattern of behaviour starting to emerge, see figure 6.11.

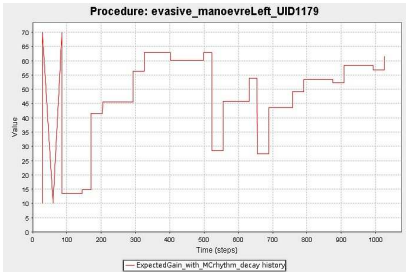
[16]In real life, probably you have had several experiences of walking through a corridor, having collision experiences with people who were coming towards you.



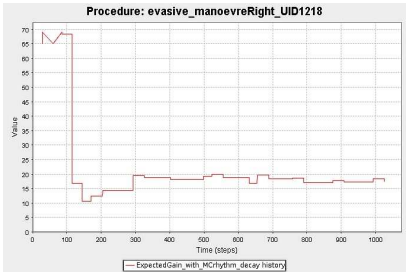
(a) Actor1: Successes, Failures Left



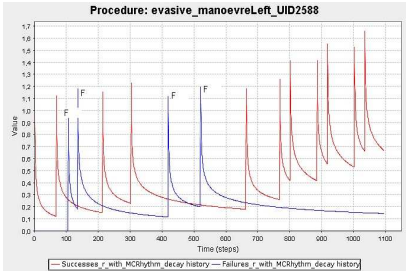
(b) Actor1: Successes, Failures Right



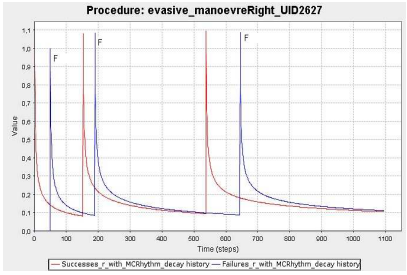
(c) Actor1: Utility Left



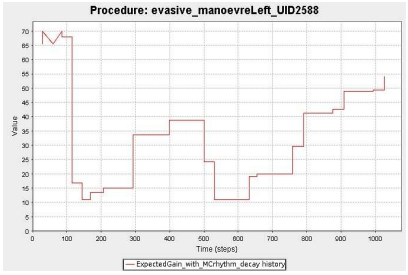
(d) Actor1: Utility Right



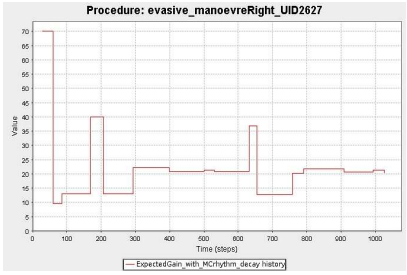
(e) Actor2: Successes, Failures Left



(f) Actor2: Successes, Failures Right



(g) Actor2: Utility Left



(h) Actor2: Utility Right

Figure 6.10: Actor1 & 2: Utility, Successes, Failures of procedure 'Left' & 'Right'.

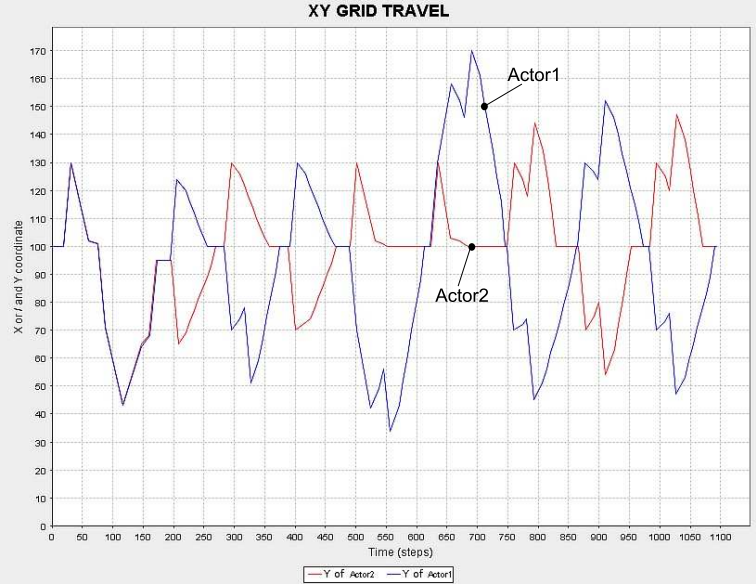


Figure 6.11: Y coordinates of both actors per time step[17].

At approximately t = 700, the pattern starts to become stable and repeats itself over and over again. As an observer from the outside, the behaviour creates the assumption that the actors have created an agreement. However, such a mutual agreement is based on many interactions and reinforcement of the successful strategy and exists not as a formal agreement on paper. Instead, it exists inside the head of each actor, i.e. the actors adapt to their environment based on their own individual experience.

6.2.2.4 Discussion

In the discussion, we want to elaborate that the behaviour in this experiment is contributing to the emergence of a social construct. Due to the stable context, the behaviour of the actors has evolved into a (tacit) social construct, 'obeyed' by both actors. Assuming that a social construct has evolved, the properties of the (tacit) social construct, see chapter 3, can be analysed as follows:

Attached norms or rules: the social construct that is formed, is based on one enforced rule, choosing the successful strategy; in this case left.

Written/unwritten: in the experiment, the rule is unwritten, but 'internalised' in the memory of the actor. However, the "traffic law" that has emerged, can easily be written down and that is exactly what most countries have

[17]The graph shows the **y path** of two actors; the application is able to plot also the *x path*, but the y path is sufficient. The moment the paths are overlapping, the actors choose a different strategy and have conflicts because they follow the same y-path. When the paths are each others opposite, or mirrored, then the actors choose the same strategy and no conflicts arise.

done in the world. Although the step of forming an externalised written social construct is not simulated in this dissertation, the next experiment will handle a social construct that is more explicit, in the sense that the social construct is communicated by a 'police' actor to the other actor.

Life span: the life span in the simulation is infinite for the construct that survives. With a more dynamic field, with entering and exiting actors, we expect that the life-span can have its limits the moment other actors with other preferences enter the field.

Roles and identification: the roles in the simulation are for both actors similar; one actor has no other privileges or role than the other actor.

Authority, responsibility and control: authority, responsibility and control are not formally present. Authority in this experiment could be seen as being shared in the community. Actors that are present in the system for a long time have authority by enforcing their behaviour to others and make them behave like they do. In this experiment, there is no authority like a policeman fining actors for incorrect behaviour, but there is some kind of indirect control. As long as the actors are passing each other successful, they do not need to correct the other. But suppose that one actor is deviating from the strategy, then the other actor(s) will immediately cause that actor to fail. A new process of "negotiation" will take place and probably the strategy that was the habit for a long time will probably 'overrule' the deviating actor.

Inheritance or prerequisite of other social constructs: there is no inheritance of other constructs, we only can say that the social construct inherits its norm from internal evolving productions.

Scenario (establishment/change or cancellation): because the social construct has to develop itself, there is not a pre-defined scenario the actors are aware of.

Context: clearly, the social construct develops and becomes active only during the time the actor perceives another actor, i.e. as long as there is a context present with actors, the social construct can reinforce itself.

Hence, the social construct that is formed during the simulation is in this case a social construct (self-regulating), that is unwritten, no roles are assigned, based on interactive behaviour, and is valid as long as the context does not change.

With this experiment, we have demonstrated that, as a first point, RBot as part of MRS is capable of more than only modelling and simulating a single cognitive plausible actor. RBot is able to simulate interaction between cognitive plausible actors. Apart from that, it is possible to inspect the behaviour of actors at the individual level and the social level (as an observer).

The second point we have demonstrated is that RBot is able to simulate an informal social construct that can emerge out of interaction between actors. The actor's experience is stored at the sub-symbolic level of the procedures that register the punishment (failures) and rewards (successes) experienced by these

procedures. These experiences of choosing a strategy lead to the selection of a preferred procedure that is applicable and successful for the current environment. Based on the presence of properties of the social construct and the frequent use of a certain strategy, we can conclude that there is a form of social construct that is internalised in the mind of the actors. Its existence, in this case, is based on interactive behaviour and cognitive (sub-symbolic) learning.

The third point we have demonstrated is that organisational learning in this experiment is a form of (situated) interactive learning. Actors adapt based on the anticipation of other actors' actions and try to impose their behaviour onto others by signalling their behaviour.

In the fourth place, the number of interactions within a community has to be high enough to be able to enforce a certain type of behaviour, because actors will forget what (not) successful behaviour was when there is no interaction.

A last point to mention is that a lock-in can occur. The simulation shows that actors, after enough times of reinforcement, create a mindset that becomes resistant to change. The only way to change this resistance, is probably a dramatic shock such as entrance of actors that have a different mindset, actors that have (social) power, or the exiting of experienced actors

In the next experiment, we want to address the role of the more formally defined social construct that coordinates behaviour of actors in such a way that behaviour is regulated by authority of one actor over the other actor.

6.2.3 Experiment 3: Social construct as coordination mechanism

The previous experiment was mainly based on the interaction of RBot actors that make decisions based on their individual preferences, i.e. they make their decisions based on the principles of the ACT-R theory. This experiment will not only show the effect of interaction between cognitive plausible actors, but it will also show the working of the extension we have designed for RBot. The social construct extension makes clear what the effect of a formalised social construct is on the behaviour of the actors at the cognitive and social level. We refer back to chapter 1, in which we asked the following question:

> **2.3 Is a social construct a coordination mechanism that can influence the behaviour of interacting related actors towards a certain desired behaviour?**

In this section, we will demonstrate this by implementing an experiment that shows the effect a social construct can have on the behaviour of actors.

In chapter 3, the theory of social constructs explained the impact of social constructs on the reasoning of an actor in a 'social environment'.For instance, social constructs can have impact on for example norms, priority of goals, utility of procedures, are sensitive to beliefs and desires the actor has, and can have impact on actions of the actor. This experiment will handle only how social constructs affect the utility of procedures[18], and thereby the conflict resolution

[18]The demonstration of the influence on procedures should be sufficient to speculate how social constructs can also influence other parts of the cognitive engine

of RBot.

In the following figure, figure 6.12, we refresh the mind of the reader first by giving an overview of how social constructs influence the conflict resolution of the cognitive engine. The figure shows that social constructs 'listen' to perception (and possible other) changes. If a norm (as part of a social construct) is triggered by such a change, then it can affect the utilities of procedures of RBot. By listening to (social) changes in perception, social constructs and their attached norms respond automatically to changes in the environment, i.e. RBot is extended with norms that are socially constructed and influence behaviour immediately.

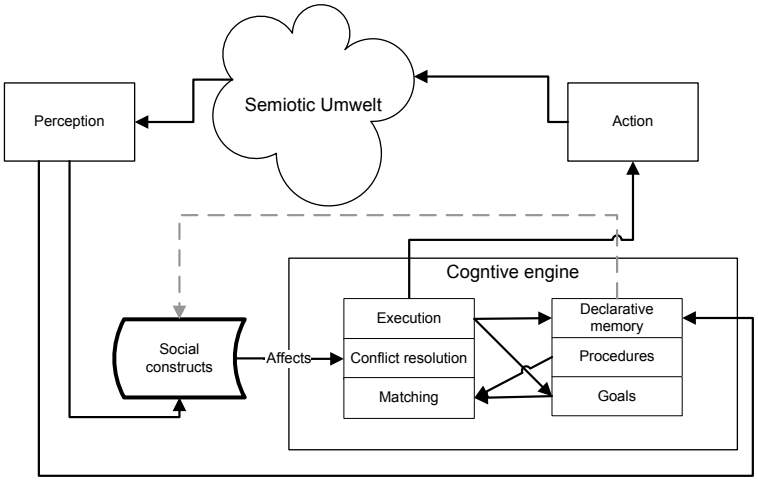


Figure 6.12: The extension of the cognitive engine with social constructs.

The previous experiment elaborated that actors adapt their behaviour because of their individual experiences. That situation is uncontrolled, in the sense that many outcomes are possible. Actors respond in their own way, i.e. they behave successfully as an individual and do not respond to the will of a (social) collective. Therefore, in this experiment, we will demonstrate that the social construct can serve as a coordination mechanism that controls the interaction between actors and results in stable *controlled* behaviour.

6.2.3.1 Description of the experiment

Experiment two already gave a description of the environment of the experiment. In the description of this experiment, the discussion is focused on the difference and addition of the social construct as a coordination mechanism in the simulation model.

The original cognitive actor, as in experiment 1 and 2, has a reasoning process that solves a problem by defining a problem space and stating clear goals. Because this process is more or less decoupled from the environment, it responds not instantaneously to changes that occur in the environment.

However, unconscious perceived aspects can influence the decision of the actor (un)willingly. For example when we drive a car, we don't reason why we drive right all the time; it is something that is socially constructed by a collective of actors in the environment. Such a social construction is added as social constructs to the cognitive architecture of RBot. To keep the awareness of the social construct up to date, a police-officer can be assigned the task of controller (socially empowered by a society) in order to correct behaviour of actors that do not behave according the society's norms or rules.

In this experiment, the social construct is triggered by the perception of the actor and affects or influences the selection of procedures as the outcome of conflict resolution. The social construct that becomes active in a certain context, can prevent or stimulate the execution of certain procedures that are linked to the construct. The experiment differs from the previous experiment in the sense that one of the actors has authority over the other. In our case, one actor (Actor2) acts as a policeman and makes the other actor (Actor1) aware of the rules when it does not behave conform the social construct as defined by the society. Actor2 is capable of doing this by sending a message; a message to the other actor that holds the social construct. When Actor1 receives the message, it parses the message and puts it in its social construct memorymap.

In the setting of the experiment, the policeman has two social constructs in its repertoire.

1. The knowledge of evading to the right side as stated by the law of, for instance, the Netherlands.

2. Communicating the knowledge to the other actor when it does not behave according the norm.

This is defined by the following XML document:
The first social construct tells us that whenever an 'Actor' is perceived, the 'rightsidelane_norm' becomes active and has as target the procedure 'evasive_manoevreRight' and motivates it with a factor 10. The second social construct is the norm that when an 'Actor' is in the perception for longer than 15 time steps, the policeman sends the first social construct of 'rightlanedriving' towards the other 'Actor'. Hence, after a collision, the policeman has still the two initial norms and the other actor received only the social construct of 'rightlanedriving' and has no social construct of sending the social construct to others.

The experiment is kept simple and only demonstrates the impact of a social norm on the behaviour of interacting actors. Therefore, the following assumptions have been made:

1. The actors are defined as being aware of their role: (1) Actor2 as policeman and (2) Actor1 obeying the policeman.

2. No negotiation takes place over the announcement of roles; the roles and relations are predefined by society.

3. Punishment is not included; however, it can be argued that by making the other actor aware of a certain social construct or norm is a correction of the behaviour. This can be regarded as a light form of punishment.

```
<socialconstruct name="rightlanedriving">
 <norm name="rightsidelane_norm">
  <normcondition name="rightsidelane_condition" type="Actor">
   <memorymap name="perception" triggertime="0"/>
  </normcondition>
  <normaction name="motivate_evasive_right">
   <affect name="evasive_manoevreRight"
           target="procedure" motivation="10"/>
</normaction>
 </norm>
</socialconstruct>
<socialconstruct name="send_rightlanedriving">
 <norm name="send_rightsidelane_norm">
  <normcondition name="send_rightsidelane_condition" type="Actor">
   <memorymap name="perception" triggertime="15"/>
  </normcondition>
  <normaction name="send_motivate_evasive_right">
   <send object="socialconstruct" name="rightlanedriving"
         target="Actor"/>
  </normaction>
 </norm>
</socialconstruct>
```

Table 6.3: Social constructs (XML format).

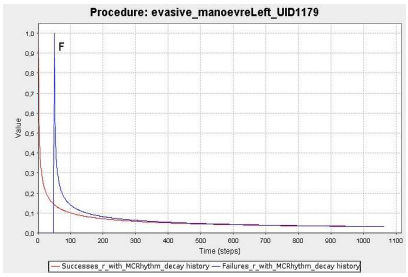
Many other assumptions can be made and in the case of the experiment, these are kept constant and therefore neglected.

6.2.3.2 Results

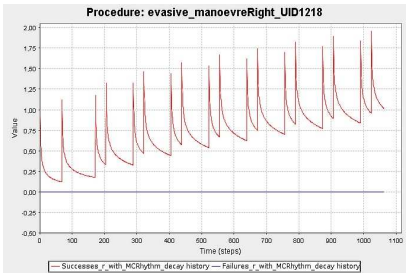
The expected results of the experiment are that when a collision takes place, the actor that does not conform to the social construct of the society, is corrected and made aware of that by the policeman.

First, we look at the individual and its productions. Figure 6.13—(a) till (h)—show some interesting results when we compare them with the findings of the previous experiment. We see for example that Actor1 is making once a failure by trying to pass on the left side, see figure 6.13a. However, in figure 6.13d, we see a correction in the utility of the procedure that promotes passing the right side. This correction is approximately a factor 10 and shows that Actor1 has become aware of the social norm 'rightlanedriving'. The police man (Actor2) has a high preference for selecting the procedure of 'rightlanedriving', because it is already equipped with the social construct that motivates its 'rightlanedriving' procedure. We also see that Actor2 experienced two failures by trying to pass right. This has a strong impact on the utility of 'rightlanedriving' but the procedure is persistent enough to keep Actor2 to drive at the right side of the road, i.e. a drop in the utility of the 'rightlanedriving' can be noticed. However, after forgetting the failures, it soon recovers to its previous state.

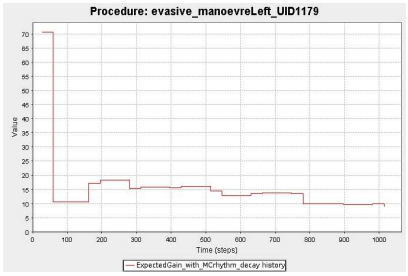
Looking at the behaviour of the actors at the social level (figure 6.14) shows



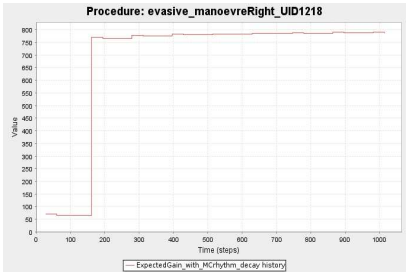
(a) Actor1: Successes, Failures Left



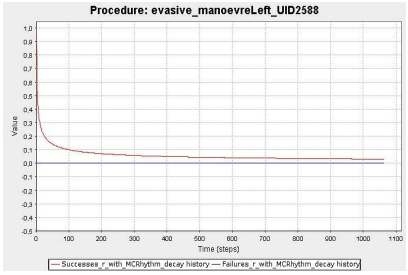
(b) Actor1: Successes, Failures Right



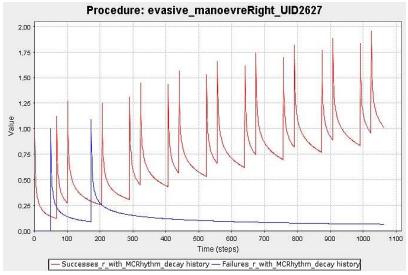
(c) Actor1: Utility Left



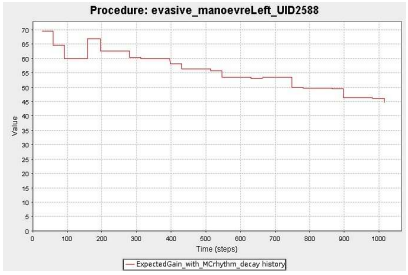
(d) Actor1: Utility Right



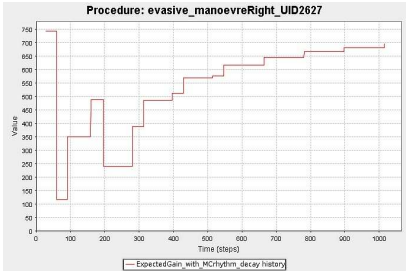
(e) Actor2: Successes, Failures Left



(f) Actor2: Successes, Failures Right



(g) Actor2: Utility Left



(h) Actor2: Utility Right

Figure 6.13: Actor1 & 2: Utility, Successes, Failures of procedure 'Left' & 'Right'.

us also that Actor1 is corrected in its behaviour at approximately time step 40 and is forced by the other actor (policeman) to choose the 'rightlanedriving' procedure. The influence of the social construct is strong and serves as a coordination mechanism for the system. After one encounter, the behaviour settles into a stable pattern.

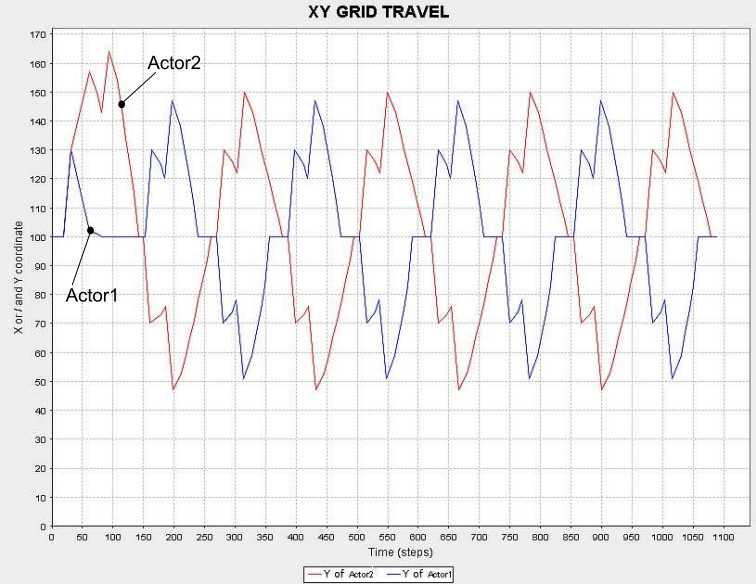


Figure 6.14: Y coordinates of both actors per timestep[19].

In the next section, we will discuss what type of social construct we have implemented. In the conclusion of this chapter, we will compare all three experiments and explain how they differ from each other, and how RBot, MRS and social constructs have created a socially and physically situated actor.

6.2.3.3 Discussion

The social construct that is applied in the experiment has demonstrated that the social construct serves as a coordination mechanism by directing and instructing actors—that show incorrect behaviour—what the desired behaviour is in the context of interaction between actors.

The properties of the social construct are different compared with the previous experiment. The formal representation of a social construct has the following properties.

Attached norms or rules: the social construct is defined by society and is enforced with the help of a policeman; 'rightlanedriving' is the norm. The

[19]For the reader: be aware that the y coordinates without the x coordinates do not say anything about the direction (right or left) the actor is moving.

other social construct is specific for policemen, i.e. the 'send_rightlanedriving' construct.

Written/unwritten: the norm is written in a traffic law; it is communicated by a policeman to the rest of the society in case the actor is not aware of the law/social norm.

Life span: the life span is dictated by the law of society. As long as the society defines that driving at the right side is the norm, the norm is propagated by actors who communicate and practise the law.

Authority: authority is given to the policeman and allows the policeman to give order and stability to the system.

Inheritance or prerequisite of other social constructs: the construct in the simulation is not inherited from other social constructs[20] .

Scenario: there is no plan or scenario involved in constructing the social norm.

Context: The traffic law or social norm is only active during the situation for which the norm is defined. This is the situation when another actor is in range of the actor. If the actor has no knowledge (or has forgotten) about the existence of the norm that is applicable to that situation, then the policeman (according to his 'send_rightlanedriving' norm) sends the social construct (as a message) containing the norm ('rightlanedriving') to the actor that does not behave according to the 'rightlanedriving' norm.

Roles and identification: the role of the policeman is clearly defined by society; the policeman has the authority to decide what is right and what is wrong.

Hence, the social construct is formal and defined by society and therefore its impact on the behaviour is strong. Because the policeman is socially empowered to dictate the norms to others, there is no negotiation possible about the correct strategy as in the second experiment.

6.3 Discussion and Conclusion

In this section, we want to discuss the difference between RBot, the ACT-R architecture and social simulations based on the results of the experiments. Secondly, we want to state that the social construct implemented in RBot is a recommendable way of bridging the gap that still exists between cognitive and social science.

The order in presentation of the experiments in this chapter demonstrates the bottom-up approach of designing the actor as elaborated in the previous chapter, i.e. starting from the cognitive actor, followed by the implementation of an environment in which the actor can 'live', interact and exchange social constructs, allowing the actor to be influenced and influence the behaviour of other

[20] Of course, there needs to be a prerequisite of the (norm-)receiving actor to understand (the language of) the policeman that 'sends the norm'.

actors. The first experiment addresses the cognitive plausibility of the actor. As elaborated in chapter 4, the cognitive plausibility of an actor is important.

> After all, social interaction is the result of individual cognition (which includes instinct, reflexes, and routines, as well as high-level conceptual processes). Therefore, details of individual cognition cannot be ignored in study sociocultural processes. (Sun, 2002, p. 214)

The RBot vs. ACT-R experiment shows a glimpse of the mechanisms that are active under the hood of RBot. The purpose of the experiment is to show that the actor RBot can be considered as a cognitive plausible model because it has inherited the mechanisms and properties of ACT-R. Cognitive plausible, to say, when running the model without the extensions that are created for making RBot interact in our MRS environment (physical situated) and the addition of social constructs and communication that allow the actor to behave socially in that environment.

The second experiment demonstrates the possibility of RBot to 'live' in an environment and interact with other RBot actors. The difference with the first experiment is the possibility of actors to perceive the actions of other actors and act upon those actions, and possess representations of itself and of others in space[21], supported by the physical memorymap as explained in chapter 5. The experiment shows that adaptation of the actor is possible by learning from interactions. Although the actors do learn from others, they actually learn from their own perceptions of the environment. In other words, the perception of other actors influence the behaviour of the actor, but the characteristics of the learning mechanisms of the individual actor still depend on the cognitive composition of the actor itself, i.e. the actor acts autonomously and determines its own actions based on its own goals, its perception apparatus, the knowledge contained in declarative and procedural memory and the cognitive mechanisms that are attached to those components. The results of the experiment have demonstrated that stability in a system can be reached (at the inter individual level) with help of perception and learning mechanisms (at the intra individual level), i.e. cognitive learning mechanisms and perception contribute to the emergence of a stable situation at the level of interactions[22].

In the third experiment, the RBot actor not only lives in an environment (MRS) and is capable to perceive the actions of other actors, but also the RBot actor's capabilities, in comparison with the RBot actor in experiment 1 and 2, have changed. We have added the concept of social constructs and message passing between actors as explained in chapter 3 and 5. The social construct allows the actor to build up representations of others, with the difference that these representations are not limited to assumptions about others, but also allow actors to communicate their intentions to other actors and integrate the communicated intentions of others in their cognitive system.

[21] Only the other actors perceived physical presence, *not* the other actors (social) beliefs, intentions, and so on.

[22] We will not call this a social level, because the actor does not take into account the needs of others.

When we compare the behaviour with that of experiment 2, we notice that the behaviour is similar. The difference is that the behaviour is directed by social constructs (norms) and that the strategy chosen will quickly settle to the 'rightlanedriving' behaviour. In experiment 2, there are no representations of social constructs that dictate which side (left or right) is preferable over the other, i.e. experiment 2 is based on individual preferences. Hence, in experiment 3, we see a quick correction in behaviour of both actors choosing the 'rightlanedriving' strategy. The influence of the social construct is that Actor1 who adopted the social norm now becomes social aware of what is desirable in the society controlled by the policeman (Actor2) who coordinates and controls the appropriate behaviour. We see at the cognitive level that the utility for 'rightlanedriving' is enforced heavily and compared with experiment 2, there is no doubt that Actor1 is choosing this production. At the social or from an observers point of view, we see an immediate punishment of the policeman forcing Actor1 to choose 'rightlanedriving', i.e. the policeman forces Actor1 to follow the norm immediately. In experiment 2, the struggle of choosing a side is a random process compared to the behaviour in this experiment. Therefore, the effect of the social construct creates a strong commitment, i.e. the actor behaves according the social construct/norm in the traffic law experiment.

The conclusion of the experiments is that we have demonstrated three important issues concerning our implementation questions.

1. Experiment 1 has shown that RBot is capable of solving problems in a similar way as ACT-R. Therefore, we can state that RBot implements the ideas of ACT-R and can be categorised among other architectures as 'cognitive plausible'. The cognitive plausibility of RBot does not only rest on experiment 1, but (eventually) on the implementation of all components and equations of ACT-R (with a few exceptions: the innovations). These innovations are functionalities equivalent to the characteristics of ACT-R. Hence, we acknowledge that a severe cognitive testing program of RBot is still necessary to show its potential.

2. Experiment 2 demonstrated that an environment[23] allows RBot to build up representations of perceived actions of other actors and is capable to incorporate those representations in its decision making. Compared to ACT-R, the ability to create an environment that supports communication and interaction between actors allows actors to be physically situated. Moreover, the experiment has shown that we can inspect such behaviour, i.e. not only the experience of a single actor can be monitored, but also the complete system of interacting actors that adapt their behaviour to the environment and other actors.

3. The last experiment has shown that social constructs allow actors to consider the norms of a social system and that actors can take a role (policeman and civilian) in their interactions. By implementing a social construct

[23] Actually, the environment is a 'semiotic Umwelt', because the environment sends signals to the actors, similar as the example of the tick explained in chapter 3.

memorymap, the actor is equipped with social representations of what the rules of a society are and has a mechanism implemented that makes the actor aware of those representations.

To complete the conclusion, we can state that the experiments of RBot, in combination with the environment MRS, have answered our questions sufficiently to conclude that RBot can be categorised not only as a cognitive actor, but as a social actor as well. Secondly, the Multi-RBot System has proven that RBot can be plugged into a (message based) Multi-Agent System.

The next chapter is a conclusive chapter that will complete the dissertation with a summary about the contents, questions raised and answered in the preceding chapters and what conclusions can be drawn from this dissertation. We will end the dissertation in the next chapter with what further work has to be done to make RBot (even) more cognitive and social plausible.

CHAPTER 7

Discussion and Conclusions

In this chapter, we want first of all refresh the mind of the reader and explain what we think is important to be addressed in the discussion of the dissertation. For this, we will refer back to the previous chapters and especially chapter 1 that gave the direction for the research path that has been followed throughout this dissertation.

The purpose of the dissertation is to remove the blinkers from both cognitive science and social or organisational science. The dissertation made explicit that behaviour of human beings cannot solely be understood by cognitive or social science alone, i.e. this research states that the actions of the actor depend on both, the cognitive and the social or organisational context of the actor.

Secondly, we want to raise a couple of new questions; questions concerning our cognitive actor RBot, the construction of the virtual world in which they live (MRS) and the position our cognitive actor RBot takes in the cognitive and social simulation community.

The previous point addresses the need for new development and further research. This chapter will end with a section that addresses areas of research in which RBot as a cognitive agent-based social simulation toolkit can be applied and what remaining work needs to be done.

7.1 Research findings

In chapter 1, our questions guided us along a road starting from Multi-Agent Systems towards the social and cognitive actor that resulted in our new cognitive agent-based social simulation toolkit that consists of RBot and MRS.

The main research question was to investigate what the aspects of actors are that plausibly can explain social interactive behaviour. We tried to tackle this question by adopting the view of methodological individualism that states

that all social phenomena are purely based on cognitive processes of individual agents. The Multi-Agent System (MAS) methodology and techniques we adopted and explained in chapter 2 fit adequately for solving our research question. However, MAS concentrates predominantly on coordination mechanisms, emergence of behaviour and interaction between agents. Although it applies the individual actor as core unit of its methodology, most applications in the field of MAS are not that concerned with the cognitive plausibility of agents.

The notion of strong agency as applied in MAS made clear that most agents are described and implemented at the rational level of description. A description at the intentional (or rational) level assumes rational behaviour according to economic theory and perhaps boundedly rational actor behaviour. It does not take into account the functional level (see chapter 4) that incorporates cognitive mechanisms and functions that allow for better modelling of individual cognition based on which better aggregate models of social behaviour can be developed.

On the other hand, although studies that concern social cognition already exist for some time (e.g. Suchman, 1987; Vera & Simon, 1993), computational models of cognition up till now have often neglected social, organisational and cultural processes. When we develop models that attempt to explain social interactive behaviour, we need to take into account the cognitive plausibility of the actor for a better understanding of the origins of social behaviour. And when we model a cognitive actor that is able to exhibit social behaviour, we should also take into account two types of knowledge:

> One kind comes from individual sources and the other from social sources. This dichotomy may be somewhat simplistic, but the contrast between the two kinds should prove to be important. The individual sources are made up of learning processes involving an agent's interaction with its environment (both sociocultural and physical) and the resulting (implicit or explicit) representations. The other kind of knowledge involves social sources, which results from concepts and conventions formed in broader contexts and is somehow imposed on (or even "forced upon") individuals. (Sun, 2002, p. 192)

In chapter 3, we introduced the concept of social construct, an explicit (and possible implicit) representation that allows for modelling emerging and existing conventions in formal and informal organisational settings, i.e. the social construct is a knowledge entity or representation that connects the individual actor with the social world and the other way around.

In chapter 5, we created a design of the social cognitive actor RBot equipped with a cognitive architecture based on ACT-R. Next, we created a task environment (MRS) in which multiple actors can perceive and interact with each other. We also implemented the social construct at the meta-cognitive or normative level of the individual actor, enabling the actor to remember and exchange social constructs applicable to certain situations, i.e. the actor has a separate habitual or situational memory.

Compared to the distinction between social and individual sources of knowledge, we do not want to create such a sharp line between social and individual sources of knowledge. At the end, any social source of knowledge becomes an individual source of knowledge, the moment the actor understands and uses the social source of knowledge. There is always an opportunity for our RBot actor to forget or change the representation of social constructs before sharing it again with the rest of its community. Although we think that the difference is not that strict, we agree with Sun in the sense that a social construct (as a source) can have impact on implicit or explicit knowledge stored in the memory of the actor, while personal knowledge in the memory of the actor may turn into a social construct by processes of communication and negotiation.

We state that this research, and especially the concept of social construct, can contribute to cognitive science as well as the social and organisational sciences.

Cognitive science In case of cognitive science, the social construct as a knowledge construction and representation in the mind of the individual allows cognitive scientists to model social phenomena such as norms, coordination mechanisms and other types of plans that exist both in the mind of the actor and as social artefacts (e.g. signs formalised on paper) in the outside world (semiotic Umwelt).

Social and organisational sciences The social and organisational sciences aim at explaining organisational behaviour based on informal as well as formal coordination mechanisms and social structures, assuming that a behaviouristic approach of individuals is sufficient. However, the social and organisational sciences deserve a better grounding based on more sophisticated theories explaining the behaviours of individual humans. In our opinion, social constructs embedded in cognitive plausible actors can be a promising start for understanding the connection between individually constructed behaviour and socially constructed behaviour.

In the remainder of our research findings, we want first to discuss the theoretical research questions as we posed in chapter 1.

1 What are the aspects of actors that plausibly explain interactive (social) behaviour?

1.1 What type of a model can explain interactive (social) behaviour?

1.2 What is required for an actor to exhibit (stable) social behaviour?

1.3 What kind of an actor can plausibly handle signs, relations and social constructs?

These questions are primarily answered in chapters 2, 3 and 4. Based on those chapters, we will elaborate the theoretical conclusions we have drawn.

Secondly, we want to discuss the conclusions regarding the implementation of our cognitive and social actor RBot and the environment MRS that allows for multi-actor simulations. In chapter 1, we gave a set of implementation questions that addressed the working of the new cognitive architecture RBot based

on which we can draw conclusions regarding the applicability or usefulness of RBot (primarily answered in chapter 5 and 6). The questions concern the validation of RBot as a cognitive plausible actor and the working or demonstration of RBot combined with MRS as a successful instrument for studying and explaining interactive social behaviour.

2 How can (cognitive and social) plausible actors be implemented in a Multi-Agent System?

2.1 Is RBot comparable to ACT-R as a valid cognitive plausible actor?

2.2 Is it possible that social constructs and organisational behaviour can emerge from social interaction?

2.3 Is a social construct a coordination mechanism that can influence the behaviour of related actors that interact towards a certain desired behaviour?

In the next sections, section 7.1.1 and section 7.1.2, we will address the theoretical conclusions, and the implementation and experimental conclusions, respectively.

7.1.1 Theoretical conclusions

In our research, we are concerned with the constructive aspects of the individual and how those aspects, in interaction with others, result in (stable) social behaviour. We stated that there are many levels of description, varying from physical towards the social level of description. The main research question gives the incentive not only to look at the social level, i.e. what causes actors to be social and interact social, but also the relations or connections between the social level and other levels of description. More specifically: what aspects of the individual cause actors to behave social; and the other way around, how does social behaviour of actors influence the individual actor.

In other words, we are convinced that in order to explain social behaviour, a more constructive analysis of the individual (e.g. the intentional and functional level of description) is necessary to understand behaviour that emerges from interaction between individuals. This so-called methodological individualism is necessary to understand the implications individual behaviour has on organisational and social behaviour.

The theoretical question 1.1 addresses methodological individualism and multi-agent based simulation methodology also known as Multi-Agent Systems (MAS). We explained techniques commonly used in MAS and explained the constructive engineering methods or views that are applied to implement agent models. We concluded that most techniques are at the intentional level of description in explaining the behaviour of the individual. In our research, we want a more functional approach (the functional level) in modelling the behaviour of the individual. We prefer to use a cognitive architecture with functional descriptions of different components and cognitive mechanisms that are empirically validated based on research and experiences in the field of cognitive psychology. Hence, our conclusion is that MAS is an appropriate methodology,

however its agents need to be cognitive plausible in order to mimic individual (and social) behaviour as accurate as possible. We will come back to cognitive plausibility in question 1.3. We first want to discuss social behaviour and MAS, i.e. the theoretical question 1.2.

In chapter 3, we asked ourselves what is required for an actor to exhibit social (stable) behaviour. For an actor to understand the needs of others, it has to have the ability to place itself in the other, i.e. it has to be able to represent the other in its own mind in order to reason about the other's goals, beliefs etc. We adopted social constructivism as a theory that allows for explaining social behaviour in the perspective of the individual as well as the perspective of society.

We argued that organisations exist primarily in the mind of individuals (an organisation is ontologically subjective, see chapter 3). We are in need of such a strong notion because (1) we adopt methodological individualism and (2) in order for individuals to reason about social structures and behaviour, we argue that this is only possible when they are represented in the mind of individuals.

In the second place, we argue that actors live in a semiotic Umwelt that allows the actor to process, use and transfer signs. A semiotic Umwelt enables the actors to exchange signs or signals and construct an internal representation of that world (the Innenwelt). In order to make sense of signs (semiosis), support of a communication process is necessary for actors in order to (socially) interact with each other. Apart from that, we also argued that a communication process presupposes a signification system; a system that allows actors to make sense of signs received from the environment, and is necessary to produce meaningful signs to act upon that world.

The signification system allows the actor to become physically and socially situated in its environment. The physical situatedness is based on physical affordances and the experience of the individual with the physical world. On the other hand, social situatedness depends upon social affordances as representations in the mind. Those representations are social behaviour patterns that emerged out of interactions with individuals and have a clear distinctive normative character[1].

In our work, we used the concept of a social construct. A social construct is based on social affordances and can be seen as a representation of cooperation and coordination. A social construct is a relatively persistent socially shared unit of knowledge reinforced in its existence by its frequent use.

Hence, when we want to interact with the world, we have to socially construct that world in order to give meaning, as a community, to that (social) world. The social construction of the (social) world results into *external* and *internal* representations of social constructs. The individual holds the meaning of agreements, objects and so on in its mind, but on the other hand, many objects in the physical world have a social meaning as well when there are social constructs attached to them that are shared within a community. Therefore, we conclude that social constructs are represented (as signs or symbols) not only in the mind but also as signs, social (and physical) affordances in a world (the

[1]Some argue that everything is socially constructed. The interactions with (physical situated) objects are learnt by imitating (e.g. child-play) others, or being taught by others (e.g. driving a car).

semiotic Umwelt) that is socially constructed and agreed upon, based on meaningful interactions (semiosis) between individuals.

Thus, a signification system allows us to interact with the environment and exchange social constructs with other actors. However, in order for the actor to hold and manipulate signs using a signification system, the actor requires a system that can handle those signs and represent a signification system.

Research question 1.3 and chapter 4 address the need for a cognitive plausible actor that is capable of handling signs and social constructs. We introduced the cognitive architecture or physical symbol system because it allows us to model a cognitive plausible actor, which is able to hold representations (signs, social constructs) and to process or operate on those representations.

In chapter 4, we elaborated the differences between the classical approach (GOFAI), connectionism and embodied or situated cognition in cognitive science. In our search for a suitable cognitive architecture that can serve as a model for the cognitive actor, we made a comparison between the two most promising architectures in cognitive science today, i.e. ACT-R and SOAR. We took the decision to adopt ACT-R, mainly because of its sub-symbolic activation that resembles and allows to model connectionist-like networks. Apart from that, we argue that ACT-R has some shortcomings.

The first one is its lack of interaction with the environment, i.e. its physical and social situatedness is lacking. Therefore, we need to adopt embodied cognition and more specifically the principles of the subsumption architecture and a physical (or spatial) memory that enables the cognitive actor to be physically situated and react instantaneously to changes in the environment. We adopted a similar mechanism for making the actor socially situated in its environment by creating representations in the form of social constructs that make the actor aware of certain situations that trigger the (now social) actor to change its goals, actions, and so on.

The second shortcoming of ACT-R is that its focus is not on the implementation of a task environment in which more actors can interact with each other. Therefore, ACT-R's communication apparatus to exchange language and interact with other actors still has shortcomings (The ACT-R community, 2005)[2].

ACT-R is a sophisticated example of the classical approach in cognitive science. We concluded that in order to make ACT-R work as an architecture that also can explain social interactive behaviour, we have to come with a complete redesign of ACT-R software[3]. The new design is named RBot (programmed in JAVA) and follows the ACT-R theory closely, except for a few innovations. Apart from the actor redesign, we created a model of a task environment (Multi-RBot System: MRS) that should enable designers to model experiments with multiple actors interacting in a simulated physical and semiotic environment.

Summarised, we can conclude that in this dissertation we addressed 1. the usefulness of MAS models to model social interaction (research question 1.1), 2. that social constructs are necessary for actors to exhibit social behaviour (re-

[2]From a conversation with one of the developers of ACT-R (Niels Taatgen) we received this report.

[3]The ACT-R community discovered in version 5.0 that their model required a redesign into a more modular approach.

search question 1.2), and 3. that actors should be equipped with a cognitive architecture to handle representations such as social constructs (research question 1.3).

In the following section, we will elaborate the conclusions regarding the implementation and demonstrative experiments concerning the capability of RBot to explain interactive social behaviour.

7.1.2 Implementation and experimental conclusions

The first implementation question concerns the implementation of (1) a cognitive and (2) a social actor in a (3) task environment; an environment in which actors can interact and exchange knowledge. Before going into details about the experiments we conducted, we first want to shed some light on the problems we encountered with ACT-R when we tried to implement experiments in a multi-actor environment.

Implementation

The first problem (ACT-R version 4.0 and 5.0) was its inaccessibility regarding its API[4] and modularity of design[5] (in terms of software engineering). Besides the software issues concerning the design of ACT-R, we had to implement the social aspects (constructs) and a multi-actor environment as well. The combination of those aspects made us decide to create a complete new software model called RBot. RBot inherited as much as possible the characteristics of the theories of ACT-R, i.e. the cognitive part of the design is comparable to that of ACT-R. However, there are some shortcomings in ACT-R as stated by an internal report of the ACT-R community (The ACT-R community, 2005, p. 1):

> ... there are still significant holes in the ACT-R architecture that reflect its origins in the set of tasks that cognitive psychology has focused for purposes of experimental control—tasks that are typically routine, repetitive, and dispassionate. The architecture has difficulties in domains that emphasize metacognitive processing, strong emotions, and communication.

Besides that, we also conclude that other aspects—social aspects and multi-actor task environment—are not present in ACT-R and therefore require new mechanisms or representations to implement them.

In the design chapter, chapter 5, we drew the conclusion that we addressed the following in this dissertation: the meta-cognitive processing, the implementation of social constructs, and the communication (interaction between multiple actors). Strong emotions are for further research to be implemented, but we assume they can be implemented in a similar way as meta-cognitive processes.

Concerning the design of a cognitive architecture, we made a redesign and created the following (cognitive) innovations in RBot compared to ACT-R. The

[4] Application Program Interface

[5] ACT-R version 6.0 is a redesign of version 5.0 and was created for a better modular approach. However, this new version was released after we already redesigned ACT-R.

first innovation is the introduction of a generalised memory architecture consisting of chunks, links, and memorymaps (access structures) enabling a modular architecture and allowing a simplified modelling of complex semantic network structures compared to ACT-R. Another change is the ability to express productions as networks of chunks that allow for a better management and component reuse. The extension concerning meta-cognitive processing is the addition of social constructs as an architectural module (memorymap plus handler) to the memory of RBot. The social construct allows for interruption of normal problem solving and links external (social and physical) events to internal reasoning.

However, to be socially and physically situated, the actor requires a task environment in which it can live or better, a 'semiotic Umwelt' in which it can interpret and exchange signs. The semiotic Umwelt requires the actor to have a signification system or communication (language) module that allows the actor to exchange signs with other actors and the environment. RBot is equipped with two memorymaps that take care of the interpretation of signs from the outside world. The first memorymap is the physical (or spatial) memorymap that allows the actor to synchronise its (self-represented) position with the outside world and other actors. The other memorymap is the social construct memorymap that holds social constructs containing norms that are applicable to situations in the environment.

The capability of the actor to perceive an environment brings forth the need for a separate module that represents a virtual world in which actors can live, i.e. interaction between actors requires a simulated representation of reality; a virtual and semiotic world in which actors can exchange and make sense of each other and objects that are present in that world.

For our cognitive multi-actor simulation to work, we can now conclude that the following components are necessary to simulate a task environment in which actors are physically and socially situated:

A cognitive plausible architecture A cognitive plausible architecture or physical symbol system allows the actor to perceive and remember symbolic representations, and decide and act upon those representations. Because of its memory, the actor is able to learn from its experiences and adapt its behaviour successfully in a flexible way conform to the changes that are perceived from the environment.

A semiotic Umwelt A semiotic Umwelt or a (multi-actor) task environment in which the actor can live and interpret, produce or exchange signs with other actors and with its task environment. The physical environment is linked to the semiotic Umwelt, but only perceivable by the actor (mediated by the semiotic Umwelt). The first task of the semiotic Umwelt is to offer the actor signs that represent the physical environment to the actor. Secondly, the semiotic Umwelt is a precondition for the actor to be able to communicate, use language and behave socially by exchanging meaningful messages to other actors.

Social constructs/meta-cognition Social constructs and a subsumption mechanism (as discussed in chapter 5) allow for representation of norms, rep-

resentations of social structures and representations of other actors. The social construct allows the actor to be socially situated, respond to social events and include the social needs of other actors with whom the actor interacts.

These components are integrated in one combined software package consisting of the task environment MRS (Multi-RBot System) and the cognitive actor RBot. The following section explains the experiments that demonstrate some of the capabilities of RBot and MRS.

Demonstrative experiments

The first experiment we conducted compared the behaviour of RBot (component 1) with the behaviour of ACT-R. The simple addition-problem experiment demonstrated a couple of characteristics present both in RBot and ACT-R. Among them are the working of the goal stack, the flow of procedures, the learning of experiences regarding procedure execution, the base-level learning equation and the merging of chunks. We are aware of the fact that a more complete testing of RBot might be necessary, but the aim of our current research is not to do cognitive tests regarding the (precise) cognitive plausibility of RBot. Because RBot follows ACT-R closely (except for the enhanced capabilities mentioned above), it is assumed that it inherits the cognitive plausibility of ACT-R. Our purpose of the experiment is to show the way RBot handles its problem space by applying procedures that are executed based on goal-directed behaviour, similar to the way goals are handled in SOAR and ACT-R. Based on the many debugging cycles and experimentation of many functionalities in ACT-R, we can conclude that we developed a cognitive plausible architecture for our research aim. RBot is adequate and accurate enough for accomplishing our goal: the development of a simulation toolkit with plausible cognitive actors that enables us to explain social interactive behaviour.

The second experiment focuses on the construction of the semiotic Umwelt and task environment that allows the actors to perceive and interact with other actors. Based on the results of the experiment, we can conclude that the actors have the capability to adapt their behaviour based on experiences with other actors, i.e. when we inspect the sub-symbolic properties (intra-individual level) of the actor, we notice that the behaviour of the actor adapts itself to the behaviour of the other actor (its environment). Another aspect we noticed is that actors based on their repeated experiences with each other start to produce repetitive, predictable actions or habits.

These habits occur because the actors make use of their (reinforcement) learning capabilities that allow them to experience from the past what will (with a high likelihood) be successful behaviour when the situation repeats itself again. Repetitive or habitual actions are the basis for the emergence of social constructs. A similar kind of learning takes place in the explanation of skill development as discussed by Anderson (1982). ACT-R's initial skill development takes place by acquiring declarative knowledge. After this stage, through practise, procedures emerge that perform the task without the acquired declarative knowledge, resulting in faster execution of problem solving.

Therefore, in the case of our experiment, we can conclude, that the reinforcement of specific productions in social interaction situations is a kind of skill mechanism. Because of its repetitive nature, actors develop skills to adapt and select the correct choice when certain situations repeat themselves often. The habitual behaviour that is produced by actors is behaviour that is constructed and caused by interaction between actors. Besides the interpretation of the habitual behaviour of the individuals and their interaction, we can conclude that the task environment MRS and the perception of RBot are working adequately and enable us to create models of cognitive actors that perceive each other and are able to learn from their interactions (previous mentioned component 2).

Whereas the second experiment produced coordinated interactive behaviour based on individual adaptation to the environment (the other actor), the third experiment implements a (formal) social construct as part of a meta-cognitive process that allows actors to react immediately on a social situation and thus influence each other's behaviour. The experiment shows that social constructs have impact on the behaviour of the actor; in the case of this experiment, the utility of a procedure. Based on the results of this experiment, we can conclude that one actor is able to influence the behaviour of the other actor *intentionally* by sending a social construct message to the other actor. The strong point of RBot is that besides modelling declarative and procedural knowledge, i.e. what facts the actor knows and what the actor is capable to do, it is possible to model normative knowledge in the form of social constructs that says something about what the actor is capable to know or allowed to do according the rules or norms of a community in which the actor takes part. In our experiment, we see this normative behaviour when the actor that has to obey the policeman, makes an attempt to break the rules of the policeman. After receiving a normative message (a social construct) from the policeman that corrects the wrong doing of the disobeying actor, the transgressing actor immediately alters its behaviour. This is not the result of an imperative force, but of its action preferences that shifted instantly. If the preferred action cannot take place, its activation will diminish and, in due time, other action possibilities may become preferred again. Therefore, the implementation of social constructs not as imperative command but as a meta-cognitive capability has proven itself to be successful in this experiment

We argue, based on the theoretical, implementation and experimental questions that RBot and MRS are useful for modelling cognitive agent-based social simulations. In our experience with developing models and experiments for RBot we can conclude that the developed model is relatively easy to maintain and to understand due to its modular/component based design. In the next section we want to discuss a couple of new questions that will challenge us in the further development of RBot.

7.2 New questions raised

In this dissertation, we are interested in developing a computerised model that can explain social behaviour that results from interacting cognitive plausible actors. In this section, we want to raise a couple of questions that concern our architecture or model; questions that should guide us to new developments,

comparison with already existing work and further work (which is discussed in the next section). We will divide this section into three parts: (1) the cognitive RBot actor, (2) the task environment or semiotic Umwelt (MRS), and (3) questions concerning the applicability of RBot and MRS.

7.2.1 Questions concerning RBot

In case of our cognitive actor, we can raise the question of how accurate and cognitive plausible our RBot model should be. Many AI researchers are dealing with the performance and cognitive plausibility of the simulated individual in how it interacts with its environment and learns from prior knowledge (and acquired new knowledge). Experiments in cognitive psychology concerning a single individual are often conducted in a laboratory setting under controlled conditions. In our research we are especially interested in the relation between and modelling of cognitive plausibility and social plausible behaviour; a relation that is often neglected by both cognitive and social scientists (see Sun, 2006a; Conte & Castelfranchi, 1995b). Although the aim of RBot is not to become a full-fledged cognitive architecture and serve as an instrument in cognitive experiments, we strive for an as cognitive plausible actor as possible. Hence, we should ask ourselves questions regarding the cognitive mechanisms and components we want to implement in RBot and whether those questions could add to the social plausibility of the behaviour our cognitive actor will produce. The questions we are interested in are questions concerning the functional primitives and cognitive mechanisms of the individual's mind, and specifically those questions that can possibly explain social behaviour. As mentioned before, the mind can be divided in several functional parts or primitives[6] as follows:

Memory (types of memory) What types of memory (declarative: semantic, episodic, categorical or non-declarative: procedural (skill/habit), perceptual, emotional, spatial, short-term/long term, relational) are fruitful to be implemented in RBot in order to let RBot become even more cognitive and social plausible.

Memory processes and learning RBot already inherited several memory processes (base-level learning, merging) from ACT-R. However, apart from those processes, could it be useful to implement top-down learning ("proceduralization") or bottom-up learning (see Sun, 2001), because our experiments have shown that such mechanisms might be active in situational or social learning (habituation) as well?

Executive processes RBot relies on its social, task and physical context when it wants to execute an action. Most actions, connected to productions, are given as prior knowledge to the actor. Concerning the actions of RBot, we could ask ourselves whether we need more advanced planning modules that allow for a more structured way of planning. With help of introspection, RBot cannot only learn from its productions, it can also learn from

[6]We apply here the primitives as stated in the Proposer Information Pamphlet of the BICA program. Retrieved 18-06-2006: http://www.darpa.mil/ipto/solicitations/open/05-18_PIPPrint.htm.

its plans that are a combination of productions and relevant knowledge and (context) constraints that are necessary to fulfil tasks. Moreover, we have to think about questions that concern social plans; plans that depend on social constraints and are socially constructed by a group of interacting individuals.

Language/Symbolic communication The language or communication possibilities of RBot allow for the transportation of symbols and structured knowledge, such as goals, procedures, social constructs and so on. The development of language skills is not worked out in RBot, but we are aware that language development during discourse between actors is important for establishing agreements and social constructs between actors. We acknowledge the important work being done in cognitive science concerning language generation and the acquisition of language skills. However, we think that we should consider questions concerning language action, i.e. what is the contribution of language concerning coordination, its effect on emotions and other mechanisms or values of actors that influence social behaviour between actors.

Emotions Emotions can be considered as a cognitive moderator or as a separate module that controls emotions. Research has been carried out on emotions of the individual (see Breazeal & Brooks, 2004; Sloman, 1987; Wright et al., 1996), but there has been little progress in this area. Some questions that are concerned with emotions could become important for our research when emotions are used for communicating social needs (e.g. with help of an 'emotional' language that has meaning for a group of individuals) that can contribute to coordination and social behaviour.

Knowledge representation/logic and reasoning Knowledge can be represented in the mind as symbols, a set of knowledge chunks, inferences (logic), rules, scripts, neural networks, as implicit or explicit knowledge, and so on (cf. Anderson, 1984, pp. 2–3). RBot makes use of representations in the form of chunks and its reasoning is rule-based and not so much on logics. ACT-R and RBot, due to their sub-symbolic properties, are so-called hybrid architectures. We are interested in questions concerning how social knowledge is represented in the individual mind, especially knowledge about social structures and constructs that are shared by a group of actors.

Motivation and goals Right now, RBot's goals are given as prior knowledge and its motivation for solving those goals depends largely on the initial parameters it receives for the amount of time it can spend on a goal and the costs associated with fulfilling that goal. The generation and formation of goal structures and associated plans are not developed for RBot. Besides that, RBot (similar to ACT-R) has not a motivational (meta-cognitive) subsystem that maps the (internal and (social) external) motives of the actor to goals, plans or procedures, or other sub-symbolic properties. However, RBot has a meta-cognitive processing system that can be applied to create a motivational subsystem that works in parallel with the recognise-decide-act cycle and thereby can influence that cycle.

Remaining components The remaining questions concern components that are concerned with perceiving the outside world. Because these components are closely linked to the environment in which the actor lives, we will discuss perception in the next section.

7.2.2 Questions concerning MRS

MRS (Multi-RBot System) is a task environment or semiotic Umwelt that defines what signs can be perceived by actors, i.e. the environment delivers services to the actors, such as the synchronisation of time, notification of physical objects and other actors, and a communication platform allowing actors to exchange knowledge with each other. The second service or function of the environment is its graphical presentation to the researcher looking at his screen and delivering information concerning the behaviour of the actors on the screen. About both these subjects, we formulate questions that concern those services:

Service of the environment to its actors

Currently, the actor takes a passive role in perceiving its environment; the environment notifies the actor, the moment the server decides something is in perceptual range of the sensors of the actor. The question is whether a clever pro-active attention mechanisms should be implemented to make the actor less dependent from its environment.

Another question concerns the client-server patterns that we applied in our software architecture; it creates a dependency between actors and environment. We have to make a distinction between the server as communication facility between the actors and as substitute for a physical environment. In case there is no need for a simulated physical environment (because the actors live in a real physical environment or a 'real' virtual world like Internet), we see that the dependency on a single server as communication medium leads to brittleness of the multi-actor system. The moment the server fails, the clients become blind, deaf and silent (they can scream but there is no medium to transport their signs). This dependency prevents actors to act as a complete autonomous entity. The question is whether we should implement a Peer-to-Peer (PtP) pattern allowing actors to be a client and at the same time take the role of a server[7]. In the case of our experiments, central control gives an advantage by having direct control over the activities of actors. However, there are cases in which the PtP solution could be a better alternative. As an example, we can think of (1) the case in which robots equipped with RBot are being sent to an environment such as high risk environments (the Moon, Mars, volcano's, deep ocean) and we cannot afford the risk by having a single point of failure; (2) another case would be a huge amount of actors that want to communicate with the server. In that case, either more servers are required, or responsibility and more autonomy have to be given to the actor.We estimate that the modular design allows for a relatively simple extension of

[7]In a PtP environment, we can expect actors (when they are cooperative) to share their environmental experiences, thereby fulfilling the task of the server.

RBot (capable of acting in a PtP infrastructure) thereby becoming less dependent on the environment delivered by the server.

Service to the researcher in the form of data and graphical presentation
The current version of MRS allows to present graphs from a database that stores data generated by every actor in the simulation environment. The interface provides the graphical presentation of events or experiences during the actors lifetime, and the plots that present movement of both actors in a two dimensional field. The other type of presentation is a dynamic view of the movement of actors displayed on a 2D grid (operational graphics). In the case of the presentation to the researcher, we can wonder how important a more sophisticated environment will be for our research purposes. When we are concerned with the impact of buildings or other structures on the social behaviour of actors, then we should pay attention to a possible 3D presentation of the environment; a change of presentation of the environment which will probably not cause much change in the underlying software model. We will come back to this issue in the section that discusses further work.

Many more questions can be brought forward that address the design of RBot and MRS. Although these questions seem unnecessary without a good plan, we argue that they can give incentives to create new research ideas and provide us with the possibilities in what directions we can develop our current architecture and environment. These directions depend primarily on our decisions we will take in our further work and the type of applications or research we have in mind that can be supported by our toolkit.

7.2.3 Questions concerning simulation and application

As we already mentioned in chapter 6, we suggest that our toolkit can be applied in different types of research: (1) research concerning the further development of a toolkit that support (2) research that is interested in a specific social or cognitive situation (e.g. simulation of crowd behaviour, simulation of cooperation in a scientific research project, simulation of workflows, simulation of multi-actor planning and coordination). Although other types of applications fall outside the category of studying human behaviour, they are closely connected to our research. We can think of (1) (personal) software agents on the Internet, (2) robots that have to interact in a group or (3) applications in the game industry.

We argue that our simulation toolkit requires further development and that many more tests are necessary to validate the behaviour of our toolkit. In the first place, our research aims at explaining interactive social behaviour (based on cognitive plausible actors). Therefore questions that concern the application of simulation in our research should be combined with empirical research in the fields of cognitive, social and organisational science for a better understanding of human (social) behaviour. In the next section, that discusses further work, we will discuss what the next phase of our research will involve.

7.3 Further work

There are many possibilities to continue our research. Partly caused by the multi- (or inter-) disciplinary and complex nature of cognitive agent-based social simulation, which has its roots in many areas as the reader probably discovered in this dissertation. We do not want to elaborate too much about all the options and ideas for further work that are possible with a model as RBot. Our discussion will limit itself to the current ideas for the short term. There are two research projects with which we want to continue our research. The first project concerns the modelling of crowd behaviour (see Still, 2000). The second project concerns the effect of design of buildings on behaviour of groups/individual behaviour (see Mobach, 2005).

Crowd behaviour

Urbanisation and the exponential growth of our world population are the cause of dense concentrations of people. Apart from that, events such as a soccer game or demonstrations cause the cluttering of huge amounts of people. Today, there has been some progress in the management and control of crowd behaviour, but there still remain many questions unanswered; especially questions about when and why certain crowd behaviours escalate into riots and others not.

The behaviour of large crowds is often controlled by police forces or the army. Crowds are most of the time unorganised entities, in the sense that they do not have (formal) organisational structures. On the other hand, the police force or army is often formally organised into strict hierarchical structures. There are numerous of agent-based social simulation models that describe and attempt to predict crowd behaviour at a high level of description by creating a behaviouristic (stimulus-response) model based on a simplified model of the individual agent. These models (e.g. Swarm[8]) are successful in describing group behaviour and they resulted in models that give a good approximation of how people behave when they are all similar and imitate each other's behaviour.

However, when we study soccer riots, we know that at least one group is organised. In this case, the police as a group has a formal organisation and therefore behaves 'organised'. Because there is a clear difference between the crowd and the police in the way they are organised, we argue that there also should be a difference in the way they behave.

Whereas the crowd is probably impulsive and reacts on unexpected situations that can occur during crowd behaviour, the police men are trained and organised and have clear instructions how to deal with certain situations. Therefore, we can state that there is a difference in rules (of the game) and social structures between the crowd and the police force.

Many researchers define differences between individuals (agents) of these groups by giving the two groups each different attributes or rules. In our opinion, such an approach is correct when there is no need for studying the individual. However, in the case of training police officers to be part in preventing or

[8]See http://www.swarm.org/wiki/Main_Page.

controlling a riot, the behaviour of the individual according to scripts, plans and norms needs to be modelled as well.

The research conducted in this dissertation can contribute to give a better understanding and prediction of behaviour of individuals in crowds, i.e. further work concerning simulations in the domain of crowd behaviour is an aim of the near future. As a side effect, such a challenge can prove if our model can be scaled up towards a simulation that consists of many, maybe thousands of agents.

3D Organisation Science: design based on simulating behaviour of human agents

The introduction of modelling buildings in 3D virtual reality starts to pay off as a serious step in the design process of a building. The construction of buildings with help of computer graphics today becomes more and more affordable for companies with small budgets. Models that are developed with help of computers (Computer Aided Design) have found its way to almost any company that is occupied with construction projects. After the introduction of 2D CAD models, followed up by 3D (2½D) models, 3D environments in which people can see projections of 3D models are the next step.

Besides the 3D projection of buildings, architects are interested in what the impact of a building is on the behaviour of the individual as well as the behaviour of the building on groups of individuals. For the purpose of this project, it is an advantage that our research project already integrated physical affordances as a mechanism in the actor. Besides that, social constructs are modelled as well, which can provide for a better understanding of social behaviour. With help of affordances and social constructs, we are able to model actors with different characters that respond in various ways to the design of for instance corridors, light fall through windows and so on.

Hence, most multi-agent software that already exists for 3D simulations are concerned with the way the actors respond to the environment; the agents are behaviour and environment (situation) oriented similar to the embodied and situated cognition approach. What is missing in this type of simulation agents is cognition, a cognitive architecture that makes an agent more pro-active and autonomous, i.e. it transforms the agent into a cognitive plausible actor. RBot provides a cognitive architecture that enriches the reactive agents in the 3D simulations.

Whereas the focus in 3D simulations is often concentrated on graphics and the projection of the 3D environment, in RBot (and MRS), the focus is on cognition and social interaction. Clearly, there is a benefit for both parties: the 3D simulations takes advantage of the cognition in RBot, and RBot benefits from the presentation and graphics possibilities of 3D simulations.

Thus, our aim is to plug the RBot actor into a 3D environment and let the actor interact with the environment based on the perceived signs from that environment and other actors that are in that environment. The challenge of the project will be to see if we are able to plug our RBot actor in an already existing environment; an environment that will function as a replacement of our simple

MRS environment.

Remaining work

There will be many tasks to fulfil besides the projects. In parallel with the projects, we will create new experiments, tests and enhancements for our RBot actor and MRS environment.

Another aspect is that we didn't pay enough attention to the data presentation and the graphical user interface of RBot and MRS. At this moment, the number of users of RBot consists only of four users. However, when the application needs to be deployed to a larger community, then the user interface requires a lot of work for making RBot accessible and user-friendly.

From an organisational point of view, we are interested in creating simulation experiments that concentrate on coordination of tasks. The experiments fall in line with Computational & Mathematical Organization Theory, i.e. we want to test cooperation and conflicts in the distribution of tasks/goals. Although such experiments will be theoretically, they can contribute to a better understanding of an effective organisation of distribution of tasks, i.e. they allow us to create formal tests about organisation theories.

Besides some new theoretical experimentation and design improvements of RBot, probably the most important tests to conduct in case of RBot is a further external validation of RBot with the help of empirical testing and evidence, especially where innovations with respect to the already tested ACT-R architecture are concerned.

Because we agree that such testing is important, we devote our last remarks of this dissertation to the issue of empirical testing. The dissertation that is in hands of the reader is an instrument that needs to be validated with help of empirical experiments. Although we are aware that there is a lack of empirical evidence that enables the reader to test its validity, we state that digging for the right answers first requires the right spade and such a spade only comes with experience.

APPENDIX A

Addition Productions*

1	**Start-Problem push goal check declarative memory**	
	IF **AND**	- the goal is to do an addition problem - status is start and column is empty
	THEN	- set the current goal status to "after checking declarative memory" - push a new subgoal chunktype check_declarative to retrieve the complete answer to the addition from memory
2	**Check addition problem in memory**	
	IF **AND**	- the goal is to retrieve the answer from memory - the answer to the problem is found in memory
	THEN	- substitute the answer in a chunk and put it in the communication buffer - remove the subgoal - set the upper goal status to found
3	**Found and clean**	
	IF	- the goal has a status of found
	THEN	- remove the top goal *(end cycle solved problem)*
4	**After check declarative memory start problem**	
	IF	- the goal has status "after checking declarative memory
	THEN	- set goal status to start *and* column = 1
5	**Addition read goal**	
	IF	- status equals start and column == 1 Remember temporary variables col1number1 = 5 col1number2 = 7 nrColumns = 4
	THEN	- set goal status to wait for subgoal - transfer nrColumns=4 to retrievalbuffer - push subgoal chunktype add_column number1 5 number2 7 column 1 carry Zero

*Productions of the multi-column addition problem in English Language

Appendix A: Addition Productions

6	**Add numbers no carry**	
	IF	- the goal is to add two number with no carry number1 = 5 number2 = 7 carry = Zero
	AND	- if the summation fact (5 + 7=?) can be retrieved from memory
	THEN	- set the answer of add_column to the answer of the summation fact (12) - set the status(carry) on waiting
7	**Add numbers process carry**	
	IF	- the goal is to add two numbers with a carry number1 = 5 number2 = 7 carry = one
	AND	- if the summation fact (number1 + carry = newnum1 (5+1 = ?)) can be retrieved from memory
	AND	- if the summation fact (newnum1 + number2 = answer (6+7 = ?) can be retrieved from memory
	THEN	- set the answer of add_column to the answer of the summation fact(13) - set the status(carry) on waiting
8	**Add column answer greater than 9 (creating a carry)**	
	IF	- the goal is to find out if there is a carry after addition number1 = 5 number2 = 7 answer = 12 carry == waiting
	AND	- it is not the last column (column != nrColumns)
	AND	- the summation fact (10 + addend2 = answer (10 + 2 = 12) can be retrieved from memory
	THEN	- set the answer to addend2 (2) and carry to one - remove nrColumns from retrieval buffer - put current goal in retrieval buffer - pop the goal with the add_column
9	**Add column answer smaller than 10 (no carry)**	
	IF	- the goal is to find out if there is no carry after addition number1 = 2 number2 = 7 answer = 9 carry == waiting
	AND	- it is not the last column (column != nrColumns)
	AND	- the summation fact (0 + addend2 = answer (0 + 9 = 9) can be retrieved from memory
	THEN	- set the answer to addend2 (9) and carry to zero - remove nrColumns from retrieval buffer - put current goal in retrieval buffer - pop the goal with the add_column *(back at top goal == addition_problem)*
10	**Add column last column**	
	IF	the goal is to add the last column of the addition problem carry == waiting
	AND	- it is the last column (column equals nrColumns)
	THEN	- set carry to "finished" - remove nrColumns from retrieval buffer - put current goal in retrieval buffer - pop the current goal with the add_column

11	**Addition problem write column_abstract** ***(depending on column number)***	
	IF	the goal is to write the answer to the goal chunk column !=null nrColumns != column column is 1...4 (column number) col(column number)answer == null
	AND	- in the retrieval buffer is a chunk of add_column
	AND	- retrieve (1+ column number = new column number (e.g. 2 → 3) from memory
	AND	- read the new numbers of the new column
		- write the answer to the col(column number)answer slot of the goal
	THEN	- remove the chunk from retrieval buffer and put nrColumns in retrieval buffer
		- push the new subgoal with a new add_column of the new column
12	**Addition problem write last column**	
	IF	- the goal is to write the answer of the last column to the goal chunk nrColumns equals column number status equals not communicate
	AND	- if there is an add_column chunk in the retrieval buffer
		- write the answer to the last column
	THEN	- set status to communicate
		- remove the chunk from the retrieval buffer
13	**Addition problem communicate**	
	IF	- status is communicate
		- put goal chunk in communication buffer
	THEN	- put the chunk in declarative memory
		- pop the goal : problem solved

Bibliography

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann Machines. *Cognitive Science*, *9*, 147–169.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., & Angel, S. (1977). *A Pattern Language*. Oxford University Press, New York.

Alexander, J. C., & Giesen, B. (1987). From Reduction to Linkage: the Long View of the Micro-Macro Link. In Alexander, J. C., Giesen, B., Munch, R., & Smelser, N. J. (Eds.), *The Micro-Macro Link*, pp. 1–44. University of California Press, Berkeley.

Anderson, J. R. (1974). Retrieval of prepositional information from long-term memory. *Cognitive Psychology*, *6*, 451–474.

Anderson, J. R. (1976). *Language, Memory and Thought*. Erlbaum, Mahwah, NJ.

Anderson, J. R. (1982). Acquisition of skill. *Psychological Review*, *89*, 369–406.

Anderson, J. R. (1983). *The Architecture of Cognition*. Harvard University Press, Cambridge, MA.

Anderson, J. R. (1984). Correspondent's report of cognitive psychology. *Artificial Intelligence*, *23*, 1–11.

Anderson, J. R. (1987). Methodologies for studying human knowledge. *Behavioral and Brain Sciences*, *10*, 467–505.

Anderson, J. R. (1990). *The adaptive character of thought*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Anderson, J. R. (1993). *Rules of the Mind*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An Integrated Theory of Mind. *Psychological Review*, *111*(4), 1036–1060.

Anderson, J. R., Fincham, J. M., & Douglass, S. (1999). Practice and Retention: A Unifying Analysis. *Journal of Experimental Psychology*, *25*, 1120–1136.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Erlbaum, Mahwah, NJ.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Review*, *2*, 396–408.

Anderson, M. L. (2003). Embodied Cognition: A field guide. *Artificial Intelligence*, *149*, 91–130.

Andrews, E. (2003). *Conversations with Lotman: cultural semiotics in language, literature and cognition*. Toronto Studies in Semiotics and Communication. University of Toronto Press, Toronto.

Argyris, C., & Schön, D. (1978). *Organizational learning: A theory of action perspecitve*. Erlbaum, Mahwah, NJ.

Atkinson, C., & Shiffrin, M. (1968). Human memory: A proposed system and its control processes. In Spence, K. W., & Spence, J. T. (Eds.), *The Psychology of Learning and Motivation*, Vol. 2. Academic Press, New York.

Austin, J. L. (1962). *How To Do Things With Words*. Oxford University Press, Oxford.

Axelrod, R. (1984). *The Evolutution of Cooperation*. Basic Books, New York.

Axelrod, R. (1986). An Evolutionary Approach to Norms. *American Political Science Review*, *80*(4), 1095–1111.

Baddely, A. (1986). *Working memory*. Oxford University Press, London.

Balci, O. (1998). *Handbook of Simulation*, chap. 10. Verification, Valdation, and Testing, pp. 335–396. John Wiley & Sons, Inc., New York, NY.

Baranauskas, C., Liu, K., & Chong, S. (2003). *Dynamics and Change in Organizations*, chap. 3. Websites as representamina of organizational behaviour, pp. 63–88. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Barber, K. S., & Martin, C. E. (1999). Agent Autonomy: Specification, Measurement, and Dynamic Adjustment. In *Proceedings of the Autonomy Control Software Workshop at Autonomous Agents 1999 (Agents'99)*, pp. 8–15, Seattle, WA.

Barjis, J., Dietz, J. L. G., & Liu, K. (2001). *Information, organisation and technology: Studies in organisational semiotics*, chap. 8. Combining the DEMO methodology with semiotic methods in business process modelling, pp. 213–246. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Batson, C. D., & Shaw, L. L. (1991). Encouraging Words concerning the Evidence for Altruism. *Psychological Inquiry*, *2*, 159–168.

Beaudoin, L. P. (1994). *Goal Processing in Autonomousd Agents*. Ph.D. thesis, School of Computer Science: University of Birmingham.

Berger, P. L., & Luckmann, T. (1967). *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. Anchor Books, Garden City, New York.

Berger, R. C., & Calabrese, R. J. (1975). Some explorations in initial interaction and beyond: Toward a developmental theory of interpersonal communication. *Human Communication Research*, *1*, 99–112.

Best, B. J., & Lebiere, C. (2003). Spatial Plans, Communication, and Teamwork in Synthetic MOUT Agents. In *Proceedings of the 12th Conference on Behavior Representation in Modeling and Simulation*, Scottsdale, AZ.

Best, B. J., & Lebiere, C. (2006). *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, chap. 8. Cognitive Agents Interacting in Real and Virtual Worlds, pp. 186–218. Cambridge University Press, New York, NY.

Birthwistle, G. M., Dahl, O. J., Myhrhaug, B., & Nygard, K. (1973). *Simula Begin*. Auerbach, Philadelphia.

Blumer, H. (1969). *Symbolic Interactionism*. University of California Press, Berkeley, Los Angeles, California, Berkeley, Los Angeles: CA.

Bond, A. H., & Gasser, L. (1988). *Readings in distributed artificial intelligence*. Morgan Kaufmann, San Mateo, CA.

Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The unified modeling language user guide*. Addison-Wesley, Reading, MA.

Bosman, A. (1977). *Een metatheorie over het gedrag van organisaties*. Stenfert Kroese, Leiden.

Bothell, D., Byrne, M. D., Lebiere, C., & Taatgen, N. A. (2003). ACT-R 6 Proposals. In *ACT-R Workshop Proceedings*, Pitssburgh, PA.

Bratman, M. E. (1987). *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA.

Bratman, M. E. (1990). *Intentions in Communication*, chap. 2. What is intention?, pp. 15–32. MIT Press, Cambridge, MA.

Breazeal, C., & Brooks, R. A. (2004). *Who Needs Emotions: The Brain Meets the Robot*, chap. 10. Robot Emotion: A Functional Perspective, pp. 271–310. Oxford University Press, Oxford.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14–23.

Brooks, R. A. (1991a). Intelligence without reason. In *Proceedings of the 1991 International Joint Conference on Artificial intelligence*, San Mateo, CA. Morgan Kaufmann.

Brooks, R. A. (1991b). Intelligence without representation. *Artificial Intelligence*, *47*, 139–159.

Brooks, R. A. (1999). *Cambrian Intelligence*. MIT Press, Cambridge, MA.

Brooks, R. A., & Flynn, A. M. (1989). Fast, cheap and out of control: a robot invasion of the solar system. *Journal of The British Interplanetary Society*, *42*, 478–485.

Brooks, R. A., & Stein, L. A. (1994). Building brains for bodies. *Autonomous Robots*, *1*(1), 7–25.

Bühler, K. (1934). *Sprachtheorie. Die Darstellungsfunktion der Sprache*. Verlag von Gustav Fischer, Jena.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley and Sons, Chichester, UK.

Byrne, M. D., & Anderson, J. R. (1998). *The atomic components of thought*, chap. Perception and Action, pp. 167–200. Erlbaum, Mahwah, NJ.

Campione, M., Walrath, K., & Huml, A. (2001). *The Java tutorial: a short course on the basics*. Sun Microsystems, Inc., Palo Alto, California.

Carley, K. M., & Gasser, L. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chap. 7. Computational Organization Theory, pp. 299–330. MIT Press, Cambridge, MA.

Carley, K. M., Kjaer-Hansen, J., Newell, A., & Prietula, M. (1991). Plural-Soar: A Prolegomenon to Artificial Agents and Organizational Behavior. In Warglien, M., & Masuch, M. (Eds.), *Artificial Intelligence in Organization and Management Theory: Models of Distributed Activity*, pp. 87–118. Elsevier, Amsterdam, North Holland.

Carley, K. M., & Newell, A. (1994). The nature of the social agent. *Journal of Mathematical Sociology*, *19*, 221–262.

Carley, K. M., & Prietula, M. J. (1994a). *Computational Organization Theory*. Lawrence Erlbaum, Hillsdale, NJ.

Carley, K. M., & Prietula, M. J. (1994b). *Computational Organization Theory*, chap. ACTS Theory: Extending the Model of Bounded Rationality, pp. 55–87. Lawrence Erlbaum Associates, Hillsdale, NJ.

Carley, K. M., Prietula, M. J., & Lin, Z. (1998). Design Versus Cognition: The interaction of agent cognition design on organizational performance. *Journal of Artificial Societies and Social Simulation*, *1*(3).

Castelfranchi, C. (1995). Guarantees for autonomy in cognitive agent architecture. In Wooldridge, M., & Jennings, N. R. (Eds.), *Intelligent Agents: Theories, Architectures and Languages*. Springer-Verlag, Berlin.

Castelfranchi, C. (2001). The theory of social functions: challenges for computational social science and multi-agent learning. *Cognitive Systems Research*, *2*, 5–38.

Chandler, D. (2002). *Semiotics: the basics*. Routledge, Abingdon, Oxon.

Chavez, A., & Maes, P. (1996). Kasbah: an agent marketplace for buying and selling goods. In Crabtree, B., & Jennings, N. (Eds.), *Proceedings of the 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96)*, pp. 75–90, Blackpool: London, UK. The Practical Application Company Ltd.

Chemero, A., & Cordeiro, W. (2006). "Dynamical, Ecological Sub-Persons": Commentary on Susan Hurley's Consciousness in Action. Retrieved 30-6-2006, from http://host.uniroma3.it/progetti/kant/field/hurleysymp_chemero_cordeiro.htm.

Cherry, C. (1980). *On Human Communication*. The MIT Press, London.

Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press, Cambridge MA.

Clark, A. (1999). An Embodied Cognitive Science?. *Trends in Cognitive Science*, *3*(9), 345–351.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., & Stafford, J. (2002). *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, Pearson Education, Boston, MA.

Cohen, P. R., & Levesque, H. J. (1990). Intention is Choice with Commitment. *Artificial Intelligence*, *42*(2-3), 213–261.

Cohen, P. R., & Levesque, H. J. (1991). Teamwork. *Nous*, *25*(4), 487–512.

Conte, R., & Castelfranchi, C. (1995a). *Cognitive and Social Action*. UCL Press, London.

Conte, R., & Castelfranchi, C. (1995b). Understanding the functions of norms in social groups through simulation. In Gilbert, N., & Conte, R. (Eds.), *Artificial Societies: The Computer Simulation of Social life*, pp. 252–267. UCL Press, London.

Conte, R., & Gilbert, N. (1995). Introduction: computer simulation for social theory. In Conte, R., & Gilbert, N. (Eds.), *Artificial societies: the computer simulation of social life*, pp. 1–15. UCL Press, London.

Conte, R., & Paolucci, M. (2001). Intelligent Social Learning. *Journal of Artificial Societies and Social Simulation*, *4*(1).

Cowart, M. (2006). Embodied Cognition. In *The Internet Encyclopedia of Philosophy*. http://www.iep.utm.edu/e/embodcog.htm.

Cunningham, D. J. (1998). Cognition as semiosis: The role of inference. *Theory and Psychology*, *8*, 827–840.

Daston, L. (1988). *Classical probability in the Enlightenment*. Princeton University Press, Princeton, NJ.

Dautenhahn, K. (1995). Getting to know each other – Artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, *16*, 333–356.

Dautenhahn, K., Ogden, B., & Quick, T. (2002). From Embodied to Socially Embedded Agents – Implications for Interaction-Aware Robots. *Cognitive Systems Research*, *3*, 397–428.

Davis, D. N. (2001). Control States and Complete Agent Architectures. *Computational Intelligence*, *17*(4), 621–650.

Davis, E. E., & Sokolove, P. G. (1975). Temperature Responses of Antennal Receptors of the Mosquito, Aedes aegypti. *Journal of Comparative Physiology*, *96*, 223–236.

Decker, K. (1995). *Environment Centered Analysis and Design of Coordination Mechanisms*. Ph.D. thesis, Department of Computer Science, University of Massachusetts, Amherst.

Deely, J. (1990). *Basics of Semiotics*. Indiana University Press, Bloomington, IN.

Deely, J. (2001). Umwelt. *Semiotica*, *134*(1/4), 125–135.

Dennett, D. C. (1978). *Brainstorms: Philosophical essays on mind and psychology*. Harvester Press, Hassocks, Sussex.

Dennett, D. C. (1987). *The Intentional Stance*. Bradford Books / MIT Press, Cambridge, MA.

Dennett, D. C. (1991). Real patterns. *Journal of Philosophy*, *87*, 27–51.

Dignum, F. (1996). Autonomous agents and social norms. In Falcone, R., & Conte, R. (Eds.), *ICMAS'96 Workshop on Norms, Obligations and Conventions*, pp. 56–71, Kyoto, Japan.

Doran, J., & Palmer, M. (1995). The EOS project: integrating two models of paeleolithic social change. In Gilbert, N., & Conte, R. (Eds.), *Artificial Societies: the Computer Simulation of Social Life*, pp. 103–125. UCL Press, London.

Durfee, E. H. (1999). *Distributed Problem Solving and Planning*, chap. 3, pp. 121–164. Cambridge, MA: MIT Press, Cambridge, MA.

Durfee, E. H., & Lesser, V. R. (1991). Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Sytems, Man, and Cybernetics*, *21*(5), 1167–1183.

Durfee, E. H., Lesser, V. R., & Corkill, D. D. (1992). Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, *1*(1), 63–83.

Eco, U. (1976). *A Theory of Semiotics*. Indiana University Press, Bloomington, IN.

Eco, U. (1990). *The Limits of Interpretation*. Indiana University Press, Boomington, IN.

Edmonds, B. (1998). Capturing social embeddedness: A constructivist approach. *Adaptive Behavior*, *7*(3/4), 323–347.

Edwards, P. (1976). *The encyclopedia of philosophy*. MacMillan Publish Co., New York.

Epstein, J. M. (2001). Learning to be Thoughtless: Social Norms and Individual Competition. *Computational Economics*, *18*, 9–24.

Epstein, J. M., & Axtell, R. (1996). *Growing Artificial Societies–Social Science from the Bottom Up*. MIT Press, Cambridge, MA.

Erman, L., Hayes-Roth, F., Lesser, V. R., & Reddy, D. R. (1980). The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys*, *12*(2), 213–253.

Etzioni, A. (2000). Social Norms: Internalization, Persuasion, and History. *Law & Society Review*, *34*, 157–178.

Faber, N. R. (2006). *Knowledge in Sustainable Behaviour*. Ph.D. thesis, SOM Research School: University of Groningen.

Fayol, H. (1916/1918). Administration industrielle et generale. Tech. rep., Bulletin de la Sociéte de l'Industrie Minerale Dunod, Paris.

Fehr, E., & Gächter, S. (2000). Fairness and Retaliation: The Economics of Reciprocity. *Journal of Economic Perspectives*, *14*(3), 159–181.

Ferber, J. (1999). *Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence*. Addison-Wesley, Reading, MA.

Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994). KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management ((CIKM) '94)*, pp. 456–463, Gaithersburg, MD, USA. ACM Press.

FIPA (2002). FIPA ACL Message Structure Specification. Retrieved 17-5-2004, from http://www.fipa.org/specs/fipa00061/SC00061G.html.

Fiske, J. (1990). *Introduction to Communication Studies*. Routledge, London.

Fiske, S. T., & Taylor, S. F. (1991). *Social Cognition*. MacGraw-Hill, New York.

Flynn, F., & Chatman, J. (2003). What's the norm here? Social categorization as a basis for group norm development. In Polzer, J., Mannix, E., & Neale, M. (Eds.), *Research in groups and teams*, pp. 135–160. JAI Press, Elsevier Science, London.

Fodor, J. A., & Pylyshyn, Z. W. (1988). *Connections and Symbols*, chap. Connectionism and cognitive architecture: A critical analysis, pp. 3–71. MIT Press, Cambridge, MA.

Forrester, J. (1991). System Dynamics and the lessons of 35 Years. In Greene, d. K. B. (Ed.), *The Systemic Basis of Policy Making in the 1990s*, chap. 1, pp. 5–34. MIT Press, Cambridge, MA.

Frankish, K. (2006). Review of Consciousness in Action, by Susan Hurley. Cambridge, MA: Harvard University Press, 1998. Retrieved 30-06-2006, from http://www.open.ac.uk/Arts/philos/Hurley_review.pdf.

Franklin, S. (1997). Autonomous Agents as Embodied AI. *Cybernetics and Systems*, *28*, 499–520.

Franklin, S., & Graesser, A. C. (1996). Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In *Intelligent Agents III, Agent Theories, Architectures and Languages, ECAI '96 Workshop (ATAL)*, pp. 21–36, Berlin. Springer-Verlag.

Franssen, M. P. M. (1997). *Some contributions to methodological individualism in the social sciences*. Ph.D. thesis, University of Amsterdam.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Indianapolis, IN.

Gasser, L., Bragamza, C., & Herman, N. (1987). MACE: a flexible testbed for distributed AI research. In Huhns, M. (Ed.), *Distributed Artifical Intelligence*, pp. 119–152. Pitman Publishing: London and Morgan Kaufmann, San Mateo, CA.

Gazendam, H. W. M. (2001). *Information, Organisation and Technology: Studies in Organisational Semiotics*, chap. 1. Semiotics, Virtual Organizations and Information Systems, pp. 1–48. Kluwer Academic Publishers, Boston.

Gazendam, H. W. M. (2003). Models as coherent sign structures. In Gazendam, H. W. M., Jorna, R. J., & Cijsouw, R. S. (Eds.), *Dynamics and change in organizations: Studies in organizational semiotics 3*, pp. 183–213. Kluwer, Boston.

Gazendam, H. W. M. (2004). Organizational Semiotics: a state of the art report. *Semiotix*, *1*(1), Retrieved 10–03–2004, from http://www.semioticon.com/semiotix/newsletterindex.htm#henk.

Gazendam, H. W. M. (2006). *Planning in Intelligent Agent Systems*, chap. Coordination in Multi-Actor Systems, pp. 139–176. John Wiley & Sons, Hoboken, New Jersey.

Gazendam, H. W. M., & Homburg, V. M. F. (1996). Emergence of Multi-Actor Systems: Aspects of Coordination, Legitimacy and Information Management. In *Proceedings of the COST A3 Conference 'Management and New Technologies', Madrid, June 12-14, 1996*, pp. 323–327, Luxembourg. Office for Official Publications of the European Communities.

Gazendam, H. W. M., & Jorna, R. J. (1998). Theories about architecture and performance of multi-agent systems. Tech. rep., SOM research report 98A02, Groningen, NL.

Gazendam, H. W. M., Jorna, R. J., & Helmhout, J. M. (2006). Monitoring a social context based on social constructs. In *Interfacing Society, Technology and Organisations*, Campinas, Brasil. UNICAMP/IC.

Gazendam, H. W. M., & Liu, K. (2005). The evolution of organisational semiotics: A brief review of the contribution of Ronald Stamper. In Filipe, J., & Liu, K. (Eds.), *Studies in organisational semiotics*, Dordrecht. Kluwer Academic Publishers.

Gazendam, L., & Jorna, R. J. (2002). Transaction Cost Economics and Plausible Actors: A Cognitive Reappraisal..

Genesereth, M. R., & Ketchpel, S. P. (1994). Software Agents. *Communications of the ACM*, *37*(7), 48–53.

Genesereth, M. R., & Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaurfmann, Los Altos, CA.

Gibbs, J. P. (1965). Norms: The problem of definition and classification. *The American Journal of Sociology*, *70*(5), 586–594.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, MA.

Gigerenzer, G., & Goldstein, D. G. (1996). Reasoning the Fast and Frugal Way: Models of Bounded Rationality. *Psychological Review*, *103*(4), 650–669.

Gilbert, N. (2006). *When Does Social Simulation Need Cognitive Models?*, chap. 19, pp. 428–432. MIT Press, Cambridge MA.

Gilbert, N., & Troitzsch, K. G. (1999). *Simulation for the Social Scientist*. Milton Keynes: Open University Press, Milton Keynes.

Gillett, J. D. (1971). *Mosquitos*. Weidenfeld and Nicolson, London.

Goldkuhl, G. (2005). The many facets of communication – a socio-pragmatic conceptualisation for information systems studies. In *Proceedings of Communication and Coordination in Business Processes, an internal workshop in Kiruna, Sweden*, Kiruna: Sweden.

Goodman, N., & Elgin, C. Z. (1988). *Reconceptions in Philosophy and Other Arts and Sciences*. Routledge, London.

Grand, S., & Cliff, D. (1997). Creatures: Entertainment software agents with artificial life.. *Autonomous Agents and Multi-Agent Systems*, *1*(1), 39–57.

Hacking, I. (1999). *The Social Construction of What?* Harvard University Press, Cambridge, MA.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, *42*, 335–346.

Harold, E. R. (1999). *XML Bible*. Hungry Minds, inc., New York.

Hebb, D. O. (1949). *The Organization of Behavior: a neuropsychological theory*. Wiley, New York.

Helmhout, J. M., Gazendam, H. W. M., & Jorna, R. J. (2004). *Virtual, Distributed and Flexible Organisations: Studies in Organisational Semiotics*, chap. 9. Social construct and boundedly rational actors: A simulation framework, pp. 153–179. Kluwer Academic Publishers, Dordrecht.

Helmhout, J. M., Gazendam, H. W. M., & Jorna, R. J. (2005a). Emergence of Social Constructs and Organizational Behavior. In *Paper presented at the 21st EGOS colloquium*, Berlin.

Helmhout, J. M., Gazendam, H. W. M., & Jorna, R. J. (2005b). *Organisational Semiotics*, chap. The Role of Organizational Semiotics and Social Constructs in the Social Awareness of Simulated Cognitive Plausible Actors, pp. 112–127. INSTICC Press, Berlin.

Hendler, J. (2001). Agents and the semantic Web. *IEEE Intelligent Systems*, *16*(2), 30–37.

Hewitt, C. (1977). Viewing Control Structures as Patterns of Message Passing. *Artificial Intelligence*, *8*(3), 323–374.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Holland, J. H. (1986). *Machine Learning II*, chap. Escaping brittleness: The possibilities of general purpose machine learning algorithms applied to parallel rule-based systems, pp. 593–623. Morgan Kaufmann, Los Altos, CA.

Holland, J. H. (1995). *Hidden Order*. Addison Wesley, Reading, MA.

Holland, J. H. (1998). *Emergence: From Chaos to Order*. Addison-Wesley, Redwood City. California.

Homburg, V. M. F. (1999). *The Political Economy of Information Management*. Ph.D. thesis, RijksUniversiteit Groningen.

Hurley, S. (2001). Perception and Action: Alternative views. *Synthese*, *129*, 3–40.

Jakobson, R. O. (1960). *Closing Statements: Linguistics and Poetics*, pp. 350–377. MIT Press, Cambridge, MA.

Jakobson, R. O. (1971). Language in relation to other communication systems. In *Selected Writings*, Vol. 2. Mouton, The Hague.

Jennings, N. R. (1992). *Joint Intentions as a Model of Multi-Agent Cooperation in Complex Dynamic Environments*. Ph.D. thesis, Department of Electronic Engineering,University of London,.

Jennings, N. R. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, *75*, 195–240.

Jennings, N. R. (1996). Coordination Techniques for Distributed Artificial Intelligence. In O'Hare, G. M. P., & Jennings, N. R. (Eds.), *Foundations of Distributed Artificial Intelligence*, pp. 319–344. John Wiley & Sons, New York, NY.

Jennings, N. R., & Campos, J. R. (1997). Towards a Social Level Characterisation of Socially Responsible Agents. *IEEE Proceedings on Software Engineering*, *144*, 11–25.

Jennings, N. R., Faratin, P., Norman, T. J., O'Brien, P., & Odgers, B. (2000). Autonomous Agents for Business Process Management. *International Journal of Applied Artificial Intelligence*, *14*(2), 145–189.

Jennings, N. R., & Pople, J. A. (1993). Design and Implementation of ARCHON's Coordination Module. In *Proceedings Workshop on Cooperating Knowledge Based Systems*, pp. 61–82, Keele, UK.

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, *1*, 7–38.

Johnson, T. R. (1997). Control in Act-R and SOAR. In Shafto, M., & Langley, P. (Eds.), *Proceedings of the Nineteenth annual Conference of the Cognitive Science Society*, pp. 343–348, Hillsdale, NJ. Lawrence Erlbaum Associates.

Jorna, R. J. (1989). *Kennisrepresentaties en symbolen in de geest*. Ph.D. thesis, Rijksuniversiteit Groningen.

Jorna, R. J. (1990). *Knowledge representation and symbols in the mind*. Stauffenburg Verlag, Tübingen, Germany.

Jorna, R. J. (2002). Cognition, Actors, and Organizations, or Why Organizations are about Managing Knowledge. *SEED (Semiotics, Evolution, Energy, and Development)*, *4*, 63–94.

Jorna, R. J., Gazendam, H. W. M., Heesen, H. C., & Van Wezel, W. M. C. (1996). *Plannen en roosteren: Taakgericht analyseren, ontwerpen en ondersteunen*. Leiderdorp, Lansa Publishing.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: individual differences in working memory. *Psychological Review*, *99*, 122–149.

Kahneman, D., Slovic, P., & Tversky, A. (1982). *Judgement under uncertainty: Heuristics and biases*. Cambridge University Press, Cambridge, England.

Kalenka, S. (2001). *Modelling Social Interaction Attitudes in Multi-Agent Systems*. Ph.D. thesis, Department of Electronic Engineering, Queen Mary, University of London.

Kieras, D. E., & Meyer, D. E. (1997). An Overview of the EPIC Architecture for Cognition and Performance with Application to Human-Computer Interaction. *Human Computer Interaction*, *12*, 391–438.

Kirsh, D. (1991). Today the earwig, tomorrow man?. *Artificial Intelligence*, *47*, 161–184.

Kittock, J. E. (1993). Emergent Conventions and the Structure of Multi-Agent Systems. In Nadel, L., & Stein, D. (Eds.), *Proceedings of the 1993 Santa Fe Institute Complex Systems Summer School*, Santa Fe, CA. Addison-Wesley.

Kraaikamp, D. C. (2003). Het modelleren van een cognitieve agent. Master's thesis, University of Groningen.

Kuran, T. (1988). The Tenacious Past: Theories of Personal and Collective Conservation. *Journal of Economic Behavior and Organization*, *10*, 143–171.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence*, *33*, 1–64.

Law, A. M., & Kelton, W. D. (2000). *Simulation Modeling and Analysis*. McGraw-Hill, Singapore.

Lehman, J. F., Laird, J. E., & Rosenbloom, P. S. (2006). A gentle Introduction to Soar: 2006 update. Retrieved 15-01-2006, from http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf. An update to an earlier paper (published 1996).

Lenat, D. (1975). BEINGS: Knowledge as Interacting Experts. In *Proceedings of the 1975 IJCAI Conference*, pp. 126–133, Tblisi, 1975.

Lenat, D., & Brown, J. S. (1984). Why AM and Eurisko appear to Work. *Artificial Intelligence*, *23*, 269–294.

Lewis, D. (1969). *Convention: A Philosophical Study*. Harvard University Press, Cambridge, MA.

Lewis, R. L. (2001). Cognitive theory, Soar. In *International Encyclopedia of the Social and Behavioral Sciences*. Pergamon, Amsterdam.

Lindblom, J., & Ziemke, T. (2003). Social Situatedness of Natural and Artificial Intelligence: Vygotsky and Beyond. *Adaptive Behavior*, *11*, 79–96.

Liu, K. (2000). *Semiotics in information systems engineering*. Cambridge Univerisity Press, Cambridge, England.

López y López, F., Luck, M., & d'Iveron, M. (2005). A Normative Framework for Agent-Based Systems. In *1st International Symposium on Normative Multi-agent Systems (NorMAS2005)*, pp. 24–35, Hatfield, UK. The University of Hertfordshire.

Luck, M., McBurney, P., & Preist, C. (2004). Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent-Based Computing). Tech. rep., AgentLink, Liverpool & Southampton.

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, *37*(7), 31–40.

Mahoney, P., & Sanchirico, C. (2001). Competing Norms and Social Evolution: Is the Fittest Norm Efficient?. Tech. rep., University of Virginia, School of Law: working paper, No. 00-15, Virginia.

Malone, T. W., & Crowston, K. (1991). Towards an interdisciplinary theory of coordination. Tech. rep., MIT Sloan School of Management: Center for Coordination Science Technical Report 120, Cambridge, MA.

March, J. G. (1991). Exploration and Exploitation in Organizational Learning. *Organization Science*, *2*(1), 71–87.

March, J. G., Schulz, M., & Zhou, X. (2000). *The Dynamics of Rules: Change in written Organizational Codes*. Stanford University Press, Stanford, CA.

March, J. G., & Simon, H. A. (1958). *Organizations*. J.Wiley, New York.

Marr, D. (1982). *Vision*. Freeman, New York.

McAdams, R. H. (1997). The Origin, Development, and Regulation of Norms. *Michigan Law Review*, *96*, 338–433.

Mead, G. H. (1934). *Mind, Self and Society from the Perspective of a Social Behaviorist*. University of Chicago, Chicago.

Meckler, M., & Baillie, J. (2003). The Truth About Social Construction in Administrative Science. *Journal of Management Inquiry*, *12*(3), 273–284.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: I Basic mechanisms. *Psychological Review*, *104*(1), 3–65.

Mietus, D. M. (1994). *Understanding planning for effective decision support*. Ph.D. thesis, Faculty of Management and Organization: University of Groningen.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*, 81–97.

Minar, N., Burkhart, R., Langton, C., & Askenazi, M. (1996). The Swarm System: A Toolkit for Building Multi-Agent Simulations. Tech. rep. 96-06-042, Santa Fe Institute, Santa Fe, CA.

Minsky, M. (1985). *The Society of Mind*. Simon & Schuster, New York.

Mitrani, L. (1982). *Simulation techniques for discrete event systems*. Cambridge University Press, Cambridge, MA.

Mitroff, I., Betz, F., Pondy, L., & Sagasti, F. (1974). On managing science in the system age: two schemes for the study of science as a whole systems phenomenon. *TIMS interfaces*, *4*(3), 46–58.

Miyashita, Y. (2004). Cognitive Memory: Cellular and Network Machineries and Their Top-Down Control. *Science*, *306*(5695), 435–440.

Mobach, M. P. (2005). Improvements in the design process of organization and building with virtual reality. In Fischer, X., & Coutelier, D. (Eds.), *Research in interactive design*. Springer, Paris.

Morgenstern, O., & Von Neumann, J. (1947). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ.

Morris, R. T. (1956). A Typology of Norms. *American Sociological Review*, *21*, 610–613.

Moss, S. (2006). *Cognition and Multi-Agent Interaction*, chap. 15. Cognitive Science and Good Social Science, pp. 393–400. MIT Press, Cambridge, MA.

Moss, S., Edmonds, B., & Wallis, S. (1997). Validation and Verification of Computational Models with Multiple Cognitive Agents. Tech. rep. 9725, Centre for Policy Modelling, Manchester, UK.

Nash, J. (1950). Equilibrium points in n-person games. In *Proceedings of the National Academy of the USA*, Vol. 36, pp. 48–49.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In Chase, W. G. (Ed.), *Visual Information Processing*. New York: Academic Press, New York.

Newell, A. (1980). Physical symbol systems. *Cognitive Science*, *4*, 135–183.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, *18*(1), 87–127.

Newell, A. (1990). *Unified theories of cognition*. Harvard University Press, Cambridge, MA.

Newell, A., & Rosenbloom, P. S. (1981). *Cognitive skills and their acquisition*, chap. 1. Mechanisms of skill acquisition and the law of practice, pp. 1–56. Lawrence Erlbaum Associates, Hillsdale, NJ.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ.

Newell, A., & Simon, H. A. (1976). Computer science as emprirical inquiry: Symbols and search. *Communications of the ACM*, *19*, 113–126.

Odell, J., Van Dyke, H., Fleischer, M., & Breuckner, S. (2002). Modeling Agents and their Environment. In Giunchiglia, F., Odell, J., & Weiss, G. (Eds.), *Agent-Oriented Software Engineering (AOSE) III*, Vol. 2585 of *Lecture Notes on Computer Science*, pp. 16–31. Springer, Berlin.

O'Hare, G. M. P., & Jennings, N. R. (1996). *Foundations of Distributed Artificial Intelligence*. Wiley, New York.

Panzarasa, P., Jennings, N. R., & Norman, T. J. (2001). Social Mental Shaping: Modelling the Impact of Sociality on the Mental States of Autonomous Agents. *Computational Intelligence*, *17*(4), 738–782.

Parunak, H. V. D. (1995). *Foundations of Distributed AI*, chap. 4. Applications of Distributed Artificial Intelligence in Industry, pp. 71–76. Wiley, New York.

Pashler, H. E. (1994). Dual-task interference in simple tasks: Data and Theory. *Psychological Bulletin*, *116*, 220–244.

Pashler, H. E. (1998). *The psychology of attention*. MIT Press, Cambridge, MA.

Pearsall, J. (2002). *Concise Oxford English Dictionary*. Oxford University Press, New York.

Peirce, C. S. (1931). *Collected Papers*, Vol. I-VIII. Harvard University Press, Cambridge, MA.

Piaget, J. (1985). *The equilibration of cognitive structures*. University of Chicago Press, Chicago, IL.

Picker, R. C. (1997). Simple Games in a Complex World: A Generative Approach to the Adoption of Norms. *University of Chicago Law Review*, *64*, 1225–1287.

Pylyshyn, Z. W. (1984). *Computation and cognition: Towards a foundation of cognitive science*. MIT Press, Cambridge, MA.

Rao, A. S., & Georgeff, M. P. (1995). BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, Menlo Park, CA. AAAI Press.

Rawls, J. (1971). *A Theory of Justice*. Oxford University Press, Oxford.

Robinson, S. (2004). *Simulation: The Practice of Model Development and Use*. Wiley, Chichester, UK.

Roest, G. B. (2004). RBot-Project: cognitie, actor architectuur(in Dutch). Master's thesis, University of Groningen: Knowledge Management.

Rubin, D. C., & Wenzel, A. E. (1996). One hundred years of forgetting: A quantitative description of retention. *Psychological Review*, *103*, 734–760.

Rumelhart, D. E., McClelland, J. L., & PDP Research Group, . (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*, Vol. 1: foundations. MIT Press, Cambridge, MA.

Rumelhart, D. E., & Norman, D. A. (1978). *Semantic factors in cognition*, chap. Accretion, tuning, and restructuring: Three modes of learning, pp. 37–53. Lawrence Erlbaum Associates, Hillsdale, NJ.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, London.

Sargent, R. G. (2000). Verfication, validation, and accreditation of simulation models. In Joines, J., Barton, R. R., Fishwick, P., & Kang, K. (Eds.), *Proceedings of the 2000 Winter Simulation Conference*, pp. 50–59, Piscataway, NJ. Institute of Electrical and Electronics Engineers.

Sawyer, R. K. (2003). Artificial Societies: Multiagent Systems and the Micro-Macro Link in Sociological Theory. *Sociological Methods & Research*, *31*(3), 325–363.

Schank, R., & Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Erlbaum Associates, Hillsdale.

Schmidt, K. (1991). *Distributed decision making: Cognitive models for cooperative work*, chap. Cooperative Work: A Conceptual Framework, pp. 75–110. John Wiley and Sons, Chichester, UK.

Schneider, D. J. (1991). Social Cognition. *Annual Review of Psychology*, *42*, 527–561.

Schumpeter, J. A. (1934). *The Theory of Economic Development*. Harvard University Press, Cambridge, MA.

Searle, J. R. (1969). *Speech Acts: an Essay in the Philosophy of Language*. Cambridge University Press, Cambridge.

Searle, J. R. (1979). *Expression and Meaning: Studies in the Theory of Speech Acts*. Cambridge University Press, Cambridge, NY.

Searle, J. R. (1995). *The Construction of Social Reality*. Penguin, London.

Sebeok, T. A. (1991). *A Sign is Just a Sign*. Indiana University Press, Bloomington, IN.

Sebeok, T. A. (1994). *Signs: An Introduction to Semiotics*. University of Toronto Press, Toronto.

Sebeok, T. A. (1996). *Origins of Language*, chap. 5. Signs, bridges, origins, pp. 89–115. Collegium Budapest, Budapest.

Shannon, C. E., & Weaver, W. (1949). *A Mathematical Model of Communication*. University of Illinois Press, Urbana.

Shoham, Y. (1993). Agent Oriented Programming. *Artificial Intelligence*, *60*, 51–92.

Shoham, Y., & Tennenholtz, M. (1992a). Emergent Conventions in Multi-Agent Systems: Initial Experimental Results and Observations. In Rich, C., Swartout, W., & Nebel, B. (Eds.), *Proceedings of the 3rd International Conference on KR&R*, pp. 225–232, San Francisco, California. Morgan Kaufmann Publishers.

Shoham, Y., & Tennenholtz, M. (1992b). On the synthesis of useful social laws for artificial agent societies. In Swartout, W. R. (Ed.), *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pp. 276–281, San Jose, CA. The AAAI Press / The MIT Press.

Shoham, Y., & Tennenholtz, M. (1995). On Social Laws for Artificial Agent Societies: Off-Line Design. *Artificial Intelligence*, *73*(1-2), 231–252.

Shuell, T. J. (1986). Cognitive Conceptions of Learning. *Review of Educational Research*, *56*(4), 411–436.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, *63*, 129–138.

Simon, H. A. (1976/1945). *Administrative behaviour: A study of decision-making processes in administrative organization (3rd edition)*. The Free Press, New York.

Simon, H. A. (1982). *Models of bounded rationality*. MIT Press, Cambridge, MA.

Simon, H. A. (1996). *The sciences of the artificial (3rd ed.)*. MIT Press, Cambridge, MA.

Sismondo, S. (1993). Some Social Constructions. *Social Studies of Science*, *23*(3), 515–553.

Sloman, A. (1987). Motives, Mechanisms and Emotions. *Emotion and Cognition*, *1*, 217–234.

Sloman, A. (1993). The mind as a control system. In Hookway, C., & Peterson, D. (Eds.), *Philosopy and the Cognitive Sciences*, pp. 69–110. Cambridge University Press, Cambridge.

Sloman, A. (2001). Varieties of Affect and the CogAff Architecture Schema. In Johnson, C. (Ed.), *Proceedings Symposium on Emotion, Cognition, and Affective Computing AISB'01 Conventions*, pp. 39–48, York, UK.

Sloman, S. A. (1996). The empirical case for two systems of reasoning. *Psychological Bulletin*, *119*, 3–22.

Sloman, S. A. (1999). Cognitive Architecture. In Wilson, R., & Keil, F. (Eds.), *The MIT Encyclopedia of Cognitive Science*, pp. 124–126. MIT Press, Cambridge, MA.

Smith, R. G., & Davis, R. (1980). Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*, *11*(1), 61–70.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, *11*, 1–23.

Sperling, G. (1960). The information available in brief visual representations. *Psychological Monograph*, *74*, 1–29.

Stamper, R. K. (1973). *Information in Business and Administrative Systems*. John Wiley & Sons, New York.

Stamper, R. K. (1992). Language and computer in organised behaviour. In *Linguistic Instruments in Knowledge Engineering*, pp. 143–163. Elsevier Science, Amsterdam.

Stamper, R. K. (2001). Organisational semiotics: Informatics without the computer?. In Liu, K., Clarke, R., Andersen, P. B., & Stamper, R. K. (Eds.), *Information, organisation and technology: Studies in organisational semiotics*. Kluwer Academic Publishers, Boston, MA.

Stamper, R. K., Liu, K., Hafkamp, M., & Ades, Y. (2000). Understanding Roles of Signs and Norms in Organisations. *Journal of Behaviour and Information Technology*, *19*(1), 15–27.

Steels, L. (1994). The Artificial Life Roots of Artificial Intelligence. *Artificial Life Journal*, *1*, 75–110.

Still, K. (2000). *Crowd Dynamics*. Ph.D. thesis, Mathematics Department: Warwick University.

Stillings, N. A., Weisler, S. W., Chase, C. H., Feinstein, M. H., Garfield, J. L., & Rissland, E. L. (1995). *Cognitive Science: An Introduction*. MIT Press, Cambridge MA.

Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication*. Cambridge University Press, New York.

Sun, R. (2001). Cognitive science meets multi-agent systems: a prolegomenon. *Cognitive Systems Research*, *14*(1), 5–28.

Sun, R. (2002). *Duality of Mind*. Lawrence Erlbaum Associates, Mahwah, NJ.

Sun, R. (2003). A tutorial on CLARION 5.0. Tech. rep., RPI, New York.

Sun, R. (2006a). *Cognition and Multi-Agent Interaction*. Cambridge University Press, Cambridge, MA.

Sun, R. (2006b). *Prolegomena to Integrating Cognitive Modeling and Social Simulation*, chap. 1, pp. 3–26. MIT Press, Cambridge MA.

Sun, R., Zhang, X., & Mathews, R. (2006). Modeling Meta-Cognition in a Cognitive Architecture. In Press.

Sunstein, C. R. (1996). Social Norms and Social Roles. *Columbia Law Review*, *96*(4), 903–968.

Sycara, K. (1998). Multiagent Systems. *AI Magazine*, *19*(2), 79–92.

Taatgen, N. A. (1999). *Learning without Limits: From problem solving towards a unified theory of learning*. Ph.D. thesis, University of Groningen: Faculty PPSW, Groningen, NL.

Taatgen, N. A. (2005). Modeling parallelization and flexibility improvements in skill acquisition from dual tasks to complex dynamic skills. *Cognitive Science*, *29*(3), 421–455.

Taatgen, N. A., & Anderson, J. R. (2006). *Handbook of Computational Cognitive Modeling*, chap. In press. Constraints in Cognitive Architectures. Cambridge University Press, Cambridge, UK.

Taatgen, N. A., & Lee, F. J. (2003). Production Compilation: A simple mechanism to model Complex Skill Acquisition. *Human Factors*, *45*(1), 61–76.

Tenbrunsel, A. E., Galvin, T. L., Neale, M. A., & Bazerman, M. H. (1996). Cognitions in Organizations. In Clegg, S. R., Hardy, C., & Nord, W. R. (Eds.), *Handbook of Organizations*, chap. 2.4, pp. 311–377. Sage Publications, London.

The ACT-R community, . (2005). Research report concerning the DARPA BICA (Biologically Inspired Cognitive Architectures) program. Tech. rep., ACT-R: Carnegie Mellon University, Pitssburgh.

Troitzsch, K. G. (1997). Social science simulation – Origin, prospects, purposes. In Conte, R., Hegselmann, R., & Terna, P. (Eds.), *Simulating social phenomena*, pp. 41–54. Springer, Heidelberg.

Tulving, E. (1972). Episodic and semantic memory. In Tulving, E., & Donaldson, W. (Eds.), *Organisation of memory*, pp. 381–403, New York. Academic Press.

Tulving, E. (1985). How many memory systems are there?. *American Psychologist*, *40*(4), 385–398.

Van den Broek, H. (2001). *On Agent Cooperation: The relevance of cognitive plausibility for multiagent simulation models of organizations*. Ph.D. thesis, SOM Graduate School / Research Institute : Systems, Organization and Management.

Van Heusden, B., & Jorna, R. J. (2001). Toward a semiotic theory of cognitive dynamics in organizations. In Liu, K., Clarke, R. J., Andersen, P. B., &

Stamper, R. K. (Eds.), *Information, Organisation and Technology*, Vol. 1 of *Information and Organization Design Series*, chap. 4. Springer-Verlag, Berlin.

Vera, A. H., & Simon, H. A. (1993). Situated Action: A Symbolic Interpretation. *Cognitive Science*, *17*(1), 7–48.

Verhagen, H. J. E. (2000). *Norm Autonomous Agents*. Ph.D. thesis, The Royal Institute of Technology and Stockholm University.

Verhagen, H. J. E., & Masuch, M. (1994). *Computational Organization Theory*, chap. 3. TASCCS: A Synthesis of Double-AISS and Plural-SOAR, pp. 39–54. Lawrence Erlbaum Associates, Hillsdale, NJ.

Vogt, P. (2002). The physical symbol grounding problem. *Cognitive Systems Research*, *3*(3), 429–457.

Volterra, V. (1926). Variation and fluctuations of the number of individuals of animal species living together. In Chapman, R. N. (Ed.), *Animal Ecology*, pp. 409–448. McGraw-Hill (1931), New York.

Von Hayek, F. A. (1979). *Law, legislation, and liberty: The political order of a free people.*, Vol. 3. Routledge & Kegan Paul, London.

Von Martial, F. (1990). Interaction among autonomous planning agents.. In Demazeau, Y., & Müller, J. P. (Eds.), *Decentralized AI - Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pp. 105–120, Amsterdam. Elsevier.

Von Uexküll, J. (1934). *Streifzüge durch die Umwelten von Tieren und Menschen*. Springer Verlag, Berlin.

Von Uexküll, J. (1957). A stroll through the worlds of animals and men – a picture book of invisible worlds. In *Instinctive Behavior - The Development of a Modern Concept*. International Universities Press, New York.

Vygotsky, L. S. (1962). *Thought and Language*. MIT Press, Cambridge, MA.

Vygotsky, L. S. (1978). *Mind in Society*. Harvard University Press, Cambridge, MA.

Watkins, C. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, University of Cambridge,England.

Whitner, R. B., & Balci, O. (1989). Guidelines for Selecting and Using Simulation Model Verification Techniques. In Macnair, E. A., Musselman, K. J., & Heidelberger, P. (Eds.), *Proceedings of the 1989 Winter Simulation Conference*, pp. 559–568, Piscataway, NJ. IEEE.

Wilson, E. O. (1975). *Sociobiology: The New Synthesis*. Harvard University Press, Cambridge, MA.

Wittig, T. (1992). *ARCHON: An Architecture for Multi-Agent Systems*. Ellis Horwood, Chichester, UK.

Wooldridge, M. (1994). Coherent social action. In Cohn, A. G. (Ed.), *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, pp. 279–283, Amsterdam, The Netherlands. John Wiley & Sons.

Wooldridge, M. (1999). Intelligent Agents. In Weiss, G. (Ed.), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chap. 1, pp. 27–78. MIT Press, Cambridge, MA.

Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press, Cambridge, MA.

Wooldridge, M. (2002). *An introduction to Multi-Agent Systems*. Joh Wiley & Sons Ltd., West Sussex, England.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, *10*(2), 115–152.

Woolf, B., & Brown, K. (2002). Patterns of System Integration with Enterprise Messaging. In *Proceedings of the 9th Conference on Pattern Language of Programs 2002*. IBM.

Wright, I., Sloman, A., & Beaudoin, L. P. (1996). Towards a design-based analysis of emotional episodes. *Philosophy, Psychiatry and Psychology*, *3*(2), 101–126.

Young, H. P. (1993). The evolution of conventions. *Econometrica*, *61*, 57–84.

Young, H. P. (1995). The economics of convention. *Journal of Economic Perspectives*, *10*, 105–122.

Zdun, U., & Avgeriou, P. (2005). Architectural patterns revisited - a pattern language. In *Proceedings of 10th European Conference on Pattern Languages of Programs (EuroPlop 2005)*, pp. 1–39, Irsee, Germany.

Ziemke, T., & Sharkey, N. E. (2001). A stroll through the worlds of robots and animals: Applying Jakob von Uexküll's theory of meaning to adaptive robots and artificial life. *Semiotica*, *134*(1-4), 701–746.

Author Index

A

Abelson, R., 88, 89, 96
Ackley, D. H., 131
Ades, Y., 89
Alexander, C., 161
Alexander, J. C., 2
Anderson, J. R., 3, 5, 13, 24, 28, 29, 35, 61, 98, 103, 104, 110, 112, 113, 115–117, 119, 121–133, 135–143, 147, 152–155, 167, 173, 177, 207, 216, 218, 219, 221, 249, 252
Anderson, M. L., 106, 107
Andrews, E., 75
Angel, S., 161
Argyris, C., 91
Askenazi, M., 60
Atkinson, C., 28
Austin, J. L., 79
Avgeriou, P., 162, 164, 165, 167
Axelrod, R., 49, 50, 96, 223
Axtell, R., 10

B

Bühler, K., 75
Bachmann, F., 162
Baddely, A., 29
Baillie, J., 70
Balci, O., 9, 210, 213
Baranauskas, C., 81
Barber, K. S., 6
Barjis, J., 88
Bass, L., 162
Batson, C. D., 63
Bazerman, M. H., 70
Beaudoin, L. P., 8, 252
Berger, P. L., 12
Berger, R. C., 63
Best, B. J., 150
Betz, F., 9
Birthwistle, G. M., 18
Blumer, H., 6, 57, 66, 68, 69
Bond, A. H., 6, 19, 20
Booch, G., 173
Bosman, A., 9
Bothell, D., 112, 113, 123, 124, 126, 127, 131, 132, 136, 137, 139, 167, 177
Bragamza, C., 56
Bratman, M. E., 23, 34, 153
Breazeal, C., 32, 252
Breuckner, S, 43
Brooks, R. A., 5, 6, 15, 32, 34, 61, 83, 106, 108, 109, 152, 180, 207, 252
Brown, J. S., 19
Brown, K., 165
Burkhart, R., 60
Buschmann, F., 161
Byrne, M. D., 112, 113, 123, 124, 126, 127, 131, 132, 136, 137, 139, 167, 177

C

Calabrese, R. J., 63

Campione, M., 170
Campos, J. R., 3, 23, 24
Carley, K. M., 4, 14, 36–40, 115, 120
Carpenter, P. A., 110, 121
Castelfranchi, C., 1, 2, 22, 24, 36, 50, 97, 251
Chandler, D., 73–75
Chase, C. H., 102, 103, 105
Chatman, J., 95
Chavez, A., 59
Chemero, A., 26, 27
Cherry, C., 81
Chomsky, N., 103
Chong, S., 81
Clark, A., 83
Clements, P., 162
Cliff, D., 59
Cohen, P. R., 53
Conte, R., 1, 2, 24, 36, 50, 57, 97, 251
Cordeiro, W., 26, 27
Corkill, D. D., 20
Cowart, M., 6
Crowston, K., 50
Cunningham, D. J., 73

D

d'Iveron, M., 96
Dahl, O. J., 18
Daston, L., 118
Dautenhahn, K., 6, 83
Davis, D. N., 33
Davis, R., 51
Decker, K., 50, 52, 86
Deely, J., 73, 76
Dennett, D. C., 3, 22, 35, 84, 103, 104, 121
Dietz, J. L. G., 88
Dignum, F., 85
Doran, J., 60
Douglass, S., 112, 113, 123, 124, 126, 127, 131, 132, 136, 137, 139, 155, 167
Durfee, E. H., 20, 51, 52

E

Eco, U., 71–77, 98
Edmonds, B., 6, 209
Edwards, P., 210
Elgin, C. Z., 66
Epstein, J. M., 10, 57, 223
Erman, L., 18
Etzioni, A., 86

F

Faber, N. R., 7
Faratin, P., 59
Fayol, H., 88
Fehr, E., 63
Feinstein, M. H., 102, 103, 105
Ferber, J., 13, 18, 19, 24, 47
Fincham, J. M., 155
Finin, T., 79–82
FIPA, 82, 151
Fiske, J., 74
Fiske, S. T., 70
Fleischer, M., 43
Flynn, A. M., 109, 180
Flynn, F., 95
Fodor, J. A., 105, 106
Forrester, J., 18
Frankish, K., 26, 27
Franklin, S., 21, 32, 33
Franssen, M. P. M., 2
Fritzson, R., 79–82

G

Gächter, S., 63
Galvin, T. L., 70
Gamma, E., 161
Garfield, J. L., 102, 103, 105
Garlan, D., 162
Gasser, L., 4, 6, 19, 20, 36, 56
Gazendam, H. W. M., 3–5, 11, 45, 46, 52, 71–73, 84–86, 89, 92, 93, 97, 115, 225
Gazendam, L., 22, 121
Genesereth, M. R., 22, 33, 179
Georgeff, M. P., 153
Gibbs, J. P., 86
Gibson, J. J., 68, 85, 89
Giesen, B., 2
Gigerenzer, G., 117, 118
Gilbert, N., 1–4, 7–10, 18, 60
Goldkuhl, G., 76, 77, 79

Goldstein, D. G., 117, 118
Goodman, N., 66
Graesser, A. C., 21, 33
Grand, S., 59

H

Hacking, I., 66
Hafkamp, M., 89
Harnad, S., 83
Harold, E. R., 179
Hayes-Roth, F., 18
Hebb, D. O., 30
Heesen, H. C., 45
Helm, R., 161
Helmhout, J. M., 3, 11, 84, 89, 115, 225
Hendler, J., 81
Herman, N., 56
Hewitt, C., 19
Hinton, G. E., 131
Holland, J. H., 30, 31, 124, 168, 216
Homburg, V. M. F., 7, 46
Huml, A., 170
Hurley, S., 25–27

I

Ishikawa, S., 161
Ivers, J., 162

J

Jacobson, I., 173
Jacobson, M., 161
Jakobson, R. O., 74, 75
Jennings, N. R., 3, 6, 17, 20, 21, 23, 24, 53–56, 58, 59, 86, 88
Johnson, R., 161
Johnson, T. R., 121
Jorna, R. J., 3, 5, 11, 22, 33, 45, 70, 71, 84, 85, 89, 115, 121, 225
Just, M. A., 110, 121

K

Kahneman, D., 118
Kalenka, S., 23
Kelton, W. D., 197, 198
Ketchpel, S. P., 22, 179
Kieras, D. E., 110, 121, 124
Kirsh, D., 107, 108
Kittock, J. E., 57
Kjaer-Hansen, J., 120
Kuran, T., 30

L

López y López, F., 96
Laird, J. E., 5, 13, 28, 31, 110, 111, 117, 119
Langton, C., 60
Law, A. M., 197, 198
Lebiere, C., 3, 5, 13, 24, 28, 29, 35, 61, 98, 110, 112, 113, 121–127, 129–133, 135–143, 147, 150, 152–154, 167, 177, 207, 216, 218, 219, 221
Lee, F. J., 137
Lehman, J. F., 5, 13, 28, 111, 117, 119
Lenat, D., 19
Lesser, V. R., 18, 20, 52
Levesque, H. J., 53
Lewis, D., 223
Lewis, R. L., 110, 111, 113
Lin, Z., 4, 14
Lindblom, J., 6, 83
Little, R., 162
Liu, K., 4, 72, 81, 85, 87–89
Luck, M., 58–60, 96
Luckmann, T., 12

M

Maes, P., 58, 59
Mahoney, P., 86
Malone, T. W., 50
March, J. G., 14, 31, 87, 90
Marr, D., 85
Martin, C. E., 6
Masuch, M., 120
Mathews, R., 32, 36
McAdams, R. H., 86, 87, 90
McBurney, P., 58–60
McClelland, J. L., 5, 105
McEntire, R., 79–82
McKay, D., 79–82
Mead, G. H., 2, 5, 6, 57, 65–67, 69
Meckler, M., 70
Meunier, R., 161
Meyer, D. E., 110, 121, 124
Mietus, D. M., 7
Miller, G. A., 29

Minar, N., 60
Minsky, M., 2
Mitrani, L., 197
Mitroff, I., 9
Miyashita, Y., 28
Mobach, M. P., 255
Morgenstern, O., 49
Morris, R. T., 86
Moss, S., 2, 209
Myhrhaug, B., 18

N

Nash, J., 50
Neale, M. A., 70
Newell, A., 3, 5, 12, 13, 23, 24, 31, 35, 37–40, 61, 83, 84, 99, 104, 110, 111, 113, 115, 116, 120, 121, 126, 151, 152, 156, 166, 207
Nilsson, N. J., 33
Nord, R., 162
Norman, D. A., 30
Norman, T. J., 59, 88
Norvig, P., 24, 28, 30, 33, 42
Nygard, K., 18

O

O'Brien, P., 59
O'Hare, G. M. P., 6
Odell, J., 43
Odgers, B., 59
Ogden, B., 6, 83

P

Palmer, M., 60
Panzarasa, P., 88
Paolucci, M., 57
Parunak, H. V. D., 59
Pashler, H. E., 122
PDP Research Group, 5, 105
Pearsall, J., 5, 31
Peirce, C. S., 3, 73, 85
Piaget, J., 30
Picker, R. C., 224
Pondy, L., 9
Pople, J. A., 54, 59
Preist, C., 58–60
Prietula, M., 120
Prietula, M. J., 4, 14, 120
Pylyshyn, Z. W., 102, 104–106, 110, 115, 116

Q

Qin,Y., 112, 113, 123, 124, 126, 127, 131, 132, 136, 137, 139, 167
Quick, T., 6, 83

R

Rao, A. S., 153
Rawls, J., 69, 92
Reddy, D. R., 18
Rissland, E. L., 102, 103, 105
Robinson, S., 9, 211, 212
Roest, G. B., 61, 167
Rohnert, H., 161
Rosenbloom, P. S., 5, 13, 28, 31, 110, 111, 117, 119, 126
Rubin, D. C., 126
Rumbaugh, J., 173
Rumelhart, D. E., 5, 30, 105
Russell, S., 24, 28, 30, 33, 42

S

Sagasti, F., 9
Sanchirico, C., 86
Sargent, R. G., 211–214
Sawyer, R. K., 2, 4, 20
Schön, D., 91
Schank, R., 88, 89, 96
Schmidt, K., 46
Schneider, D. J., 69
Schooler, L. J., 127
Schulz, M., 14, 87, 90
Schumpeter, J. A., 30
Searle, J. R., 67, 68, 79
Sebeok, T. A., 72, 73, 75, 76
Sejnowski, T. J., 131
Shannon, C. E., 75
Sharkey, N. E., 72
Shaw, L. L., 63
Shiffrin, M., 28
Shoham, Y., 13, 22, 57
Shuell, T. J., 30
Silverstein, M., 161

Simon, H. A., 5, 12, 14, 31, 35, 38, 44, 45, 66, 68, 83, 89, 110, 115, 119, 189, 242
Sismondo, S., 66
Sloman, A., 8, 32–35, 252
Sloman, S. A., 5
Slovic, P., 118
Smith, R. G., 51
Smolensky, P., 29
Sommerlad, P. and Stal, M., 161
Sperling, G., 29
Stafford, J., 162
Stamper, R. K., 72, 80, 85, 86, 88, 89
Steels, L., 5
Stein, L. A., 6
Still, K.G., 255
Stillings, N. A., 102, 103, 105
Suchman, L. A., 83, 242
Sun, R., 1–3, 5, 6, 14, 32, 36, 60, 121, 153, 154, 238, 242, 251
Sunstein, C. R., 86, 88
Sycara, K., 12, 17, 20, 58

T

Taatgen, N. A., 109, 110, 115, 119, 122, 124, 130, 132, 137, 138, 155, 177
Taylor, S. F., 70
Tenbrunsel, A. E., 70
Tennenholtz, M., 13, 57
The ACT-R community, 2005, 246, 247
Troitzsch, K. G., 1, 2, 4, 7–10, 18, 60
Tulving, E., 28, 29
Tversky, A., 118

V

Van den Broek, H., 2, 19, 20, 24, 25, 35
Van Dyke, H., 43
Van Heusden, B., 5, 11, 70
Van Wezel, W. M. C., 45
Vera, A. H., 89, 242
Verhagen, H. J. E., 24, 36, 120
Vlissides, J., 161
Vogt, P., 83
Volterra, V., 18, 24
Von Hayek, F. A., 95
Von Martial, F., 45, 46
Von Neumann, J., 49
Von Uexküll, J., 71, 72, 76, 188
Vygotsky, L. S., 6, 65, 83

W

Wallis, S., 209
Walrath, K., 170
Watkins, C., 31
Weaver, W., 75
Weisler, S. W., 102, 103, 105
Wenzel, A. E., 126
Whitner, R. B., 213
Wilson, E. O., 28
Wittig, T., 54, 61
Wooldridge, M., 6, 17, 20, 21, 24, 33–35, 49, 56–58, 60, 79, 81, 82, 105, 153
Woolf, B., 165
Wright, I., 8, 252

Y

Young, H. P., 223

Z

Zdun, U., 162, 164, 165, 167
Zhang, X., 32, 36
Zhou, X., 14, 87, 90
Ziemke, T., 6, 72, 83

Index

A

accessible environment, 32
activation, 29, 106, 113, 117, 119, 121, 125–133, 139, 163, 169, 171, 172, 180, 181, 184, 187, 191, 217, 220, 221, 250
  associative, 126
  base-level, 126, 127, 130, 133
  equation, 126
  sub-symbolic, 113, 246
  threshold, 126
actor, 99, 115, 120, 121, 150, 159, 168, 180, 198, 206, 214–216, 243, 247, 248, 254
ACTS, 120
Adaptive Control of Thought, Rational, 5, 9, 10, 13, 15, 16, 24, 28, 29, 35, 61, 98, 102, 105, 106, 110, 112–139, 142–150, 152–156, 159, 160, 163, 166–169, 175–178, 182, 186, 191, 193, 207–209, 215–222, 224, 231, 237–239, 242, 244, 246–249, 251, 252, 257, 298, 300, 301, 306, 307
affordance, 24, 26, 32, 68, 85, 89, 245, 256
  physical, 27, 85, 89, 245, 256
  social, 85, 86, 245
agent, 1, 13, 42, 58–61, 150, 152, 159, 223, 244, 256
  boundedly rational, 38, 120
  cognitive social, 1, 2, 12, 17, 61, 149, 159, 161, 209, 241, 255
  deliberative, 28, 34
  reactive, 34, 36, 44, 61, 256
  reflective, 33, 35
  reflexive, 34
Agent Communication Language, 51, 81, 82, 150, 151, 195, 207
Architecture for Cooperative Heterogeneous ON-line systems, 54, 59, 61
artificial
  society, 2, 5, 10, 11, 212, 215
Artificial Intelligence, 2, 4–6, 8, 13, 18, 21, 22, 32–34, 79, 83, 102, 106, 107, 110, 150, 223, 251
  Distributed, 6, 18–20, 48–50, 52, 150, 160
artificial task environment, 150, 151
associative activation, 126
associative memory, 29, 105, 119
augmentative cooperation, 46
autonomy, 6, 19, 20, 33, 36, 37, 43, 58, 162, 165, 166, 196, 253

B

base-level activation, 126, 127, 130, 133
base-level learning, 125, 126, 131, 136, 138, 142, 147, 149, 191, 217, 220, 222, 249, 251
behaviour

goal-directed, 22, 28, 35, 120, 249
social, 151
belief, 3, 22, 23, 29, 30, 35, 39, 47, 53, 56, 57, 70, 71, 104, 109, 115, 116, 231, 238, 245
Belief Desire Intention, 35, 153
biological level, 103
bounded rationality, 40, 44, 115, 118–120, 189
boundedly rational, 14, 38, 45, 119, 147, 242
agent, 38, 120

C

chunk, 38, 84, 86, 106, 111, 113, 115, 121, 125–134, 136, 144, 146, 147, 149, 152, 154, 169–178, 180–193, 202, 203, 208, 217, 220, 221, 225, 248, 249, 252
chunking, 35, 112, 117
chunktype, 125, 128, 169, 170, 172–179, 185, 192
CLARION, 5, 6, 9, 36, 121, 154
Client-Server, 162
CNET, 51
cognitive
actor, 13, 15, 16, 61, 63, 65, 98, 101, 102, 110, 120, 150, 157, 160, 166, 216, 222, 232, 237, 240–242, 246, 249–251
architecture, 121, 166, 168
plausible, 10–14, 70, 102, 110, 115, 120, 149, 160, 163, 166, 207, 209, 216, 230, 231, 243, 244, 246, 250, 251, 254, 256
agent, 2, 13, 14, 21, 33, 36–40, 44, 61, 63, 84, 120, 250
plausible, 4, 34, 35, 40, 61
architecture, 5, 12, 13, 15, 23, 29, 32, 34–36, 60, 61, 65, 70, 98–102, 105, 106, 109, 110, 113–115, 117, 119–121, 124, 137, 153, 156, 159, 160, 166, 167, 180, 206, 215, 216, 223, 233, 242–244, 246, 247, 251, 256
level, 2, 3, 84, 98, 150, 154, 223, 225, 239
mechanism, 3, 10, 11, 23, 35, 53, 105, 156, 160, 167, 191, 207, 216, 238, 242, 244, 251
model, 2, 16, 109, 111, 114, 137
modelling, 1, 2, 5, 223
plausibility, 120, 154, 156, 216, 238, 239, 242, 245, 249, 251
plausible, 24, 35, 61, 101, 130, 147, 149, 153, 156, 207, 238, 239, 245, 251
theories, 1, 2, 35, 156, 167
cognitive agent-based computational social simulation model, 1, 2, 12, 17, 61, 159
cognitive agent-based social simulation, 12, 61, 149, 161, 209, 241, 255
Cognitive Impenetrability, 115
cognitive science, 1–6, 15, 17, 20, 24, 27, 32, 33, 35, 64, 75, 100–102, 105, 113, 120, 156, 223, 241, 243, 246, 252
communication, 3–5, 15, 19, 21, 32, 42–46, 56, 63–65, 71, 73–82, 84, 89, 95, 98, 99
act, 75
environment, 4, 13, 15, 43, 44, 61
handler, 182
language, 79, 80
layer, 81
level, 81, 82
model, 76, 78
protocol, 51
communication and social environment, 4
Component Interaction View, 162
Computational & Mathematical Organization Theory, 4, 13, 14, 36, 257
computer simulation, 8, 210
model, 18
conflict, 19, 21, 32, 37, 39, 42, 45–50, 52, 56, 114, 145, 183, 257
handling, 46, 48
resolution, 50, 114, 122, 139, 143, 144, 146, 167, 181, 183, 187, 231–233
situation, 46
connectionism, 5, 24, 102, 105–107, 113,

133, 156, 246
continuous environment, 42
Contract-Net, 51
cooperation, 1, 6, 11, 12, 14, 33, 42, 45–50, 54, 58, 85, 154, 222, 245, 254, 257
  augmentative, 46
  debative, 46
  integrative, 46
coordination, 6, 11, 12, 33, 45, 47, 48, 50, 85, 90, 97, 113, 123, 156, 180, 224, 245, 252, 254, 257
  mechanism, 14, 16, 45, 48–50, 65, 98, 207, 209, 231, 232, 236, 242–244, 300
cultural environment, 21

D

debative cooperation, 46
declarative
  learning, 125, 132
Declarative Memory, 29
deliberative agent, 28, 34
design-based approach, 8, 209
deterministic environment, 42
discrete environment, 42
discrete-event simulation, 18, 42
Distributed Communication View, 162
Distributed Problem Solving, 19, 50, 55
  Cooperative, 51
Document Type Definition, 172, 179, 185, 201
dynamic environment, 42

E

economics, 4, 20, 49, 60, 86, 105
Embodied Cognition, 6, 44, 83, 105, 156
empirical cycle, 7
environment, 4–6, 12–15, 20–22, 24–27, 31–34, 36–40, 42–45, 50, 61, 63, 71, 85, 90, 96, 98, 102, 104, 106, 109, 111, 116, 119, 124, 127, 143, 149, 188, 196, 204, 205, 208, 224, 229, 232, 238, 239, 242, 243, 246, 248, 250, 253, 256
  accessible, 32
  continuous, 42
  cultural, 21
  deterministic, 42
  discrete, 42
  dynamic, 42
  episodic, 42
  external, 27, 37, 89
  in-accessible, 32
  inner, 44
  multi-actor, 121, 124, 194, 209, 222, 247
  observable, 104
    partially, 42
  outer, 44
  perceived, 38
  physical, 2, 13, 15, 43, 44, 61
  real-time, 149
  sequential, 42
  simulated, 13, 43, 61
  simulation, 166, 254
  social, 4, 13, 15, 21, 44, 61
  socially situated, 2
  static, 42
  task
    artificial, 150, 151
episodic environment, 42
episodic learning, 117
Episodic Memory, 29
episodic memory, 29
evolutionary learning, 31
Executive Process Interactive Control, 110, 121, 124
eXtensible Markup Language, 81, 172, 177, 179, 188, 193, 201, 205, 208, 233
external environment, 27, 37, 89

F

finite state machines, 42
First In First Out, 153, 170, 173, 182
formal organisation, 69, 255
Foundation for Intelligent Physical Agents, 82, 151, 201
functional level, 3, 22–24, 35, 64, 98, 103, 105, 121, 125, 153, 242, 244

G

game theory, 49, 50, 105, 223, 225, 226

General Principle of Rationality, 116
generalisation, 134, 137, 176–178, 186
goal, 6, 19, 23, 24, 30–32, 35–39, 44–48, 50, 52, 53, 56, 59, 63, 95, 103–105, 109–114, 116, 117, 119–122, 124, 125, 133, 135, 136, 139, 140, 143, 144, 146–149, 152–155, 177, 181–183, 185, 190, 193, 201, 208, 216, 217, 219, 222, 225, 226, 231, 232, 238, 245, 246, 249, 252, 257
  buffer, 124
  chunk, 124, 134, 145, 177–179, 185–187
  comparators, 32
  directed behaviour, 22, 28, 35, 120, 249
  feeder, 167, 182, 185, 187, 193
  generators, 32
  joint, 53, 54, 56
  module, 112, 124, 144
  organisational, 115
  social, 36, 39, 57
  stack, 16, 124, 125, 133, 135, 136, 139, 144, 146, 147, 149, 153, 167, 175, 177, 182–187, 194, 217, 219, 220, 222, 249
  value, 139, 140, 146–148, 155
goal level, 146, 147
goal-directed behaviour, 22, 28, 35, 120, 249
goalhandler, 170, 182–188, 191, 194
Good Old Fashioned AI, 34, 106, 246

H

habitual or situational memory, 242
handler, 15, 168, 172, 175, 179, 181, 182, 187, 188, 190, 191, 193, 194, 248
  goal, 170, 182–188, 191, 194
  memory state dependent, 182
  message, 190, 194, 217
  perception, 182, 188, 189, 203, 204
  position, 189
  social construct, 191, 192
  time dependent, 182
Human Computer Interaction, 43

I

in-accessible environment, 32
inner environment, 44
integrative cooperation, 46
intentional level, 3, 22–24, 35, 64, 104, 105, 121, 244
Internet, 6, 18, 19, 21, 57–59, 94, 95, 150, 253, 254
Iterated Prisoner's Dilemma, 50

J

Joint Responsibility Model, 54

K

knowledge, 3, 5, 22, 23, 28–30, 37–39, 44, 45, 50, 51, 53, 56, 57, 59, 64, 74, 79, 81, 85, 86, 90, 92, 98, 100, 103–105, 107, 110–112, 116, 117, 119, 121, 122, 125, 127, 130, 132, 137, 138, 140, 151, 152, 154, 160, 166, 167, 169, 174, 175, 179, 190, 197, 201, 207, 208, 218, 233, 237, 238, 242, 243, 245, 247, 251, 252
  declarative, 125, 137, 138, 249
  explicit, 243, 252
  normative, 250
    shared, 99
  prior, 30, 37, 90, 96, 222, 251, 252
  procedural, 133, 250
  shared, 3, 12
    normative, 99
Knowledge Interchange Format, 81, 82, 179
knowledge level, 23, 106, 117
Knowledge Level Hypothesis, 104, 116
Knowledge Query Markup Language, 80–82
Knowledge Sources, 18, 19

L

Last In First Out, 124, 170, 173, 185
learning, 2, 3, 28–31, 35, 36, 42, 64, 90, 91, 110, 111, 113, 114, 117, 119, 120, 123, 132, 133, 136, 137, 139, 140, 142, 224, 225, 231, 238, 242, 249, 251
  declarative, 125, 132

episodic, 117
evolutionary, 31
machine, 30
mechanism, 13, 32, 117, 136, 177, 238
q-learning, 31
reinforcement, 30, 117, 187
semantic, 117
sub-symbolic, 35, 115, 132, 138, 217
supervised, 30
symbolic, 30, 136, 138, 217
unsupervised, 30
level
biological, 103
goal, 146, 147
intentional, 3, 22–24, 35, 64, 104, 105, 121, 244
knowledge, 23, 106, 117
organisational, 156, 208
physical, 84, 103, 109, 121
rational, 49, 53, 103, 104, 109, 153, 223, 242
semiotic, 3, 64, 82–84, 86, 98, 99, 152
symbol, 23, 104
Long Term Memory, 28, 29, 117
long-term memory, 28, 29, 117, 125

M

machine learning, 30
memory, 28–30, 33, 34, 71, 93, 100, 106, 109, 110, 114, 117, 121, 125–128, 130, 132, 133, 144, 167–175, 180, 182, 187–189, 191, 194, 204, 206–208, 217, 218, 221, 227, 229, 243, 246, 248, 251
associative, 29, 105, 119
component, 55, 169, 171–174, 179, 180, 191
decay, 119
declarative, 16, 29, 112, 115, 117, 122, 124, 125, 131–133, 136–138, 144, 146, 147, 154, 169, 175, 184, 192, 217, 220
episodic, 29
habitual or situational, 242
long-term, 28, 29, 117, 125
model, 15, 167, 168
physical, 167, 187
procedural, 29, 122, 136, 169, 175, 238
production, 115, 117, 125
semantic, 29, 117
sensory, 29
short-term, 28, 119, 153
working, 29, 111, 112, 114, 133, 167, 183, 191
memory state dependent handlers, 182
memorymap, 154, 167, 169–171, 173–176, 180–182, 184–192, 201, 205, 208, 233, 238, 240, 248
message handler, 190, 194, 217
Message Queueing, 165
methodological individualism, 2, 3, 12, 86, 98, 241, 244, 245
Micro Simulation Models, 18
Model Social Agent, 37
Model View Controller, 165, 168
multi-actor, 99, 115, 120, 121, 150, 159, 168, 180, 198, 206, 214–216, 243, 247, 248, 254
environment, 121, 124, 194, 209, 222, 247
multi-actor system, 13, 110, 120, 121, 160, 222, 253, 300
multi-agent, 1, 13, 42, 58–61, 150, 152, 159, 223, 244, 256
simulation system, 59, 60, 162
simulations, 8
Multi-Agent System, 1–4, 6, 8, 11–15, 17–21, 23, 24, 39, 42, 43, 49, 50, 52, 55–59, 63, 70, 78, 79, 81, 82, 105, 110, 120, 149, 150, 156, 157, 160–162, 166, 198, 201, 206, 208, 240–242, 244–246, 297, 299–301, 303, 305
Multi-RBot System, 4, 10, 11, 13, 15, 61, 121, 150, 151, 156, 159, 163, 164, 166, 193, 205, 207–209, 215, 216, 222, 230, 236, 238, 240–244, 246, 249–251, 253, 254, 256, 257, 300

N

negotiation, 3, 12, 48, 50, 52, 58, 64, 69, 80, 96, 230, 233, 237, 243

norm, 31, 37, 39, 46, 50, 57, 61, 77, 78, 82, 86–90, 93–97, 100, 179–181, 192, 223, 225, 229–233, 236, 237, 239, 243, 248, 250, 256
  social, 233, 234, 237, 239
normative, 7, 14, 15, 31, 33, 36, 57, 60, 61, 97, 98, 152, 156, 207, 242, 245, 250

O

observable environment, 104
organisation, 2, 4–6, 11, 15, 20, 24, 48, 53, 64, 67, 69–71, 83–87, 90, 92–95, 97, 109, 115, 120, 156, 169, 180, 223, 245, 257
  formal, 69, 255
organisational
  behaviour, 4, 14, 115, 120, 123, 149, 222, 223, 243, 244, 300
  goal, 115
  semiotics, 75, 84, 86
organisational level, 156, 208
organisational semiotics, 3–5, 12, 15, 27, 85, 99
outer environment, 44

P

Partial Global Planning, 52
partially observable environment, 42
Peer to Peer, 162, 253, 254
perceived environment, 38
perception, 20, 24–28, 32, 55, 57, 85, 101, 107, 108, 110, 112, 113, 120–124, 133–135, 144, 149, 150, 167, 169, 171, 181, 182, 187–189, 191, 202, 203, 205, 225, 232, 233, 238, 250, 253
perception handler, 182, 188, 189, 203, 204
physical
  affordance, 27, 85, 89, 245, 256
physical and socially situated environment, 2
physical environment, 39, 43, 44, 83, 150, 163, 196, 207, 248, 253
physical level, 84, 103, 109, 121
physical memory, 167, 187
physical representation, 19
physical symbol system, 12, 23, 35, 61, 84, 98, 99, 107, 109, 113, 114, 116, 153, 156, 166, 207, 246, 248
Physical Symbol System Hypothesis, 116
physical, communication and social environment, 13, 15, 44, 61
position handler, 189
Primary Modelling System, 73
Principle of Rationality, 116
procedural
  knowledge, 133, 250
  memory, 29, 122, 136, 169, 175, 238
  sub-symbolic learning, 138
  symbolic learning, 136
Procedural Memory, 29
Procedural Reasoning System, 35
proceduralization, 137, 138, 177, 251
production memory, 115, 117, 125
psychology, 4, 8, 26, 49, 64, 102
  cognitive, 24, 100, 105, 159, 207, 216, 244, 247, 251
  social, 2, 64, 69, 99, 120, 207
Publish-Subscribe, 164, 165

Q

Q-learning, 31

R

rational level, 49, 53, 103, 104, 109, 153, 223, 242
RBot, 4, 9–11, 13, 15–17, 61, 65, 98, 102, 109, 121, 122, 137, 149, 150, 153–157, 159, 160, 163, 166–170, 172–176, 178–182, 185, 186, 188, 190, 191, 193, 194, 200–202, 205, 207–209, 215–223, 230–233, 236–244, 246–257, 300, 301, 306, 307
  Multi-RBot System, 4, 13, 61, 121, 150, 151, 159, 163, 164, 166, 208, 222, 240, 246, 249, 253, 300
reactive agent, 34, 36, 44, 61, 256
real-time environment, 149
reflective agent, 33, 35
reflexive agent, 34
regulative cycle, 7, 8, 211

reinforcement learning, 30, 117, 187
Remote Procedure Call, 165
representation, 5, 9, 11, 12, 14, 20, 22, 26, 28, 34, 35, 52, 53, 57, 64, 65, 70, 71, 73, 74, 79, 81, 83–85, 89, 90, 92, 98–102, 105–107, 109, 112, 113, 115, 116, 119, 120, 122, 151, 152, 156, 160–162, 167, 180, 187, 206, 207, 211, 212, 222, 223, 236, 238–240, 242, 243, 245–249, 252
  physical, 19

S

Secondary Modelling System, 73
semantic learning, 117
Semantic Memory, 29
semantic memory, 29, 117
semiosis, 3, 5, 73, 91, 98, 188, 245, 246
semiotic Umwelt, 27, 43, 71, 72, 188, 200, 202, 206, 208, 239, 243, 245, 246, 248, 249, 251, 253
semiotics, 3, 5, 27, 64, 65, 71–74, 79, 82, 84, 156
  level, 3, 64, 82–84, 86, 98, 99, 152
  organisational, 3–5, 12, 15, 27, 75, 84–86, 99
sensory memory, 29
sequential environment, 42
Short Term Memory, 28, 29
short-term memory, 28, 119, 153
sign, 3–5, 11–13, 64, 71–76, 80–82, 84, 85, 88, 91, 92, 94, 95, 98, 99, 101, 105, 150, 151, 153, 157, 203, 208, 216, 246, 248, 253, 256, 299
  production, 64, 65, 74–76, 84, 91
signal, 24, 30, 72, 75, 88, 94, 102, 105, 150, 151, 157, 239
signification, 71, 74–76, 81
  process, 71
  system, 64, 65, 74–77, 79, 82, 84, 91, 98, 99, 201, 202, 245, 246, 248
simulated environment, 13, 43, 61
simulated physical environment, 43
simulation, 2, 4, 7–11, 18, 43, 61, 64, 102, 109, 136, 140, 141, 163, 165, 171, 182, 189, 193–196, 202, 206, 225–227, 230, 231, 256, 257
  clock, 174, 197, 198
  computer, 8, 210
  discrete-event, 18, 42
  model, 2, 6–9, 17, 212, 213, 216, 232
  social, 1, 2, 16, 18, 237, 241, 250
    cognitive agent-based, 1, 2, 12, 17, 61, 149, 159, 161, 209, 241, 255
  software, 149
  study, 9, 209, 211, 212
  system, 58, 59
simulation environment, 166, 254
social
  affordance, 85, 86, 245
  behaviour, 2, 3, 10–12, 15–17, 23, 24, 33, 38, 42, 44, 47, 61, 63–65, 70, 95, 98, 99, 149, 151, 152, 160, 207, 216, 223, 242–247, 250–252, 254, 256, 299
  cognition, 69, 70, 242
  environment, 6, 43, 44, 61, 63, 64, 74, 83, 120, 167, 222, 231
  goal, 36, 39, 57
  level, 3, 23, 24, 50, 63, 64, 82, 84–86, 98, 99, 101, 152, 156, 180, 223, 230, 231, 234, 238, 244
  norm, 13, 85, 86, 88, 152, 181, 200, 207, 223, 233, 234, 237, 239
  psychology, 2, 64, 69, 99, 120, 207
  science, 1, 2, 8, 14, 17, 20, 237, 241
social construct, 3, 12, 14–16, 57, 61, 65–67, 75, 77, 80, 82–101, 109, 152, 156, 167, 172, 180, 181, 187, 190, 192, 201, 207, 209, 216, 222, 223, 225, 227, 229–234, 236–239, 242–250, 252, 256, 299, 300
  handler, 191, 192
social construct handler, 191, 192
social construction, 64–66, 68–70, 85, 86, 94, 245
social constructivism, 2, 12, 64, 65, 67, 70, 71, 75, 85, 98, 156, 245
social constructivist, 66, 83
  theory, 4, 6
social simulation, 1, 2, 16, 18, 237, 241, 250
  model, 1, 2, 12, 17, 61, 159

social-cultural environment, 21
sociology, 49, 64, 86, 99, 120, 207
specialisation, 46, 176, 177
State, Operator And Result, 5, 13, 24, 28, 29, 31, 35, 61, 110–115, 117–121, 145, 152, 166, 169, 186, 208, 246, 249, 300, 306
static environment, 42
sub-symbolic, 29, 113, 121, 125, 126, 144, 147, 163, 167, 178, 179, 181, 191, 205, 216, 221, 231, 249, 252
  activation, 113, 246
  learning, 35, 115, 132, 138, 217
  level, 113, 116, 117, 119–121, 133, 136, 143, 147, 149, 171, 178, 179, 187, 230
  procedural learning, 138
subsumption, 61, 105, 109, 207, 248
  architecture, 15, 34, 108, 109, 152, 180, 207, 208, 246
supervised learning, 30
Swarm, 60, 255
symbol level, 23, 104
symbolic
  learning, 30, 136, 138, 217
  procedural learning, 136

T

task environment, 13, 14, 21, 38, 39, 42–44, 114, 120, 150, 156, 159, 160, 162, 163, 165, 166, 168, 190, 195, 196, 198, 200, 202, 216, 222, 242, 246, 248, 249, 251, 253
Tertiary Modelling System, 73
time dependent handler, 182

U

Umwelt, 27, 43, 71–73, 76, 83, 98, 188, 200, 202, 206, 208, 239, 243, 245, 246, 248, 249, 251, 253
unsupervised learning, 30

V

validation, 2, 8, 9, 121, 209–214, 216, 217, 222, 244, 257
verification, 8, 9, 209–214, 216

W

working memory, 29, 111, 112, 114, 133, 167, 183, 191

Summary

Multi-Agent Systems (MAS) is a field of research that is concerned with the design, analysis and implementation of computational decision and simulation systems—a collection of actors that work in conjunction with each other—to enhance one's understanding of how people cooperate and coordinate their actions to solve problems. Upon its foundation in the mid-nineties, MAS propagated in the fields of cognitive sciences and social sciences, and was adopted, more specifically, in cognitive modelling and modelling of multi-actor interaction such as in social simulation (Conte & Castelfranchi, 1995b; Troitzsch, 1997). However, there has not been enough interaction between these two fields, and tools that try to connect both fields have not been sufficiently developed (Sun, 2006b). In our research, we will address this problem and make an attempt to bring both fields closer together.

The aim of our research is to design and implement a cognitive actor-based computational social simulation model based on a selection of social and cognitive theories in an attempt to satisfy the need for a complex cognitive-social model. The motivation for our research is twofold: (1) to reveal the constituents of MAS and the social cognitive actor (cf. Conte & Castelfranchi, 1995b) based on theoretical considerations to (2) construct a simulation model that plausibly explains the interactive social behaviour of actors in a physical and socially situated environment (cf. Gilbert & Troitzsch, 1999). The aim is to relate the behaviour of individuals (micro level) that form a group or an organisation to the behaviour and performance of a group or organisation as a whole (macro level) (Alexander & Giesen, 1987; Van den Broek, 2001).

Social (and organisational) behaviour is an outcome of the interactions between individuals. These interactions can lead to (temporarily) stable patterns of (organisational) behaviour. An organisation can be seen as a group of people that has habits of action aimed at cooperation and coordination of work. An organisation is not a physical, tangible object like an apple or a computer keyboard. Its observation, demarcation and existence depend on the existence of human habits and human-produced signs, and is a product of interactive social behaviour (Helmhout et al., 2004). An organisation can be traced back to representations (in the mind of actors) that structure the interaction among people, thereby demarcating a group of people who are members of an organisation

(Van Heusden & Jorna, 2001).

The interaction and relationship between organisation, actors and the mental representation inside the actors is evident, and therefore, (controlled) organisation is not possible without coordination among actors and their representations. In our models, we use the concept of social constructs: social constructs can be seen as representations of cooperation and coordination, based on intertwined habits and mutual commitments that are often expressed in sign structures such as agreements, contracts, plans, etc. (Liu, 2000; Gazendam, 2003, p. 205).

In our research, a mixed-level analysis (Sun, 2006b) is performed, in which the following levels are discerned in descending order:

1. The *social level*, which involves negotiation, reaching agreements, social laws and overall behaviour influenced by individual behaviour of actors.
2. The *semiotic level*, which describes the use of language and signs in communication and interaction in order to reach an agreement on social constructs (e.g. common plans, or contracts) (Gazendam, 2004; Helmhout et al., 2004, 2005b).
3. The *intentional level*, which ascribes beliefs, desires and intentions to actors. Intentions of others, inferred from knowledge about others' beliefs and desires, enable the examination of others' actions (Dennett, 1987).
4. The *functional level* describes learning and cognitive mechanisms of the actor and is grounded in an empirically validated theory (cf. Anderson & Lebiere, 1998).
5. The *physical/physiological level* is a level described by appealing to physics, biology and chemistry. It predicts or explains behaviour in terms of physical laws or physiological properties (Dennett, 1987).

The social construct has different meanings and is present in all of these levels. First of all, a social construct is the result of interaction between actors. At the social level, actors habituate their action and thereby create a habitat of social constructs. Sign production delivers codes or signs (the semiotic level) to which actors assign (shared) meaning. This shared meaning or knowledge becomes normative at the moment it influences behaviour of the members in the community, e.g. norms that guide behaviour such as respect for older people. In order to change and understand meaning, actors require a signification system (Eco, 1976). Such a signification system needs to be grounded in the actor, which can be done with the help of a plausible cognitive architecture at the cognitive level (functional level), e.g. ACT-R (Anderson & Lebiere, 1998). It enables the actor to process and produce signs. The social construct or a derivative of the social construct (sign) needs to be present at the cognitive, social and semiotic level. In the community, the social construct is present as a sign(-structure). These signs or social constructs are exchanged by actors in the community via communication and rituals. The exchange is made possible by a medium that allows the transport of social constructs as messages, documents or events. Communication between actors and the reinforcement of social constructs (caused by the frequent

use of social constructs) will lead to stable habituated patterns of behaviour and shared normative knowledge. Furthermore, the individual actor needs a physical symbol system (Newell, 1980) that enables it to hold representations, process and produce signs/symbols/social constructs and exhibit intelligent action.

We are interested in the interaction between these levels. In our research, we aim not only to describe the interaction, but also to simulate these interactions with the help of a Multi-Agent System. The combination of theory and MAS has lead to the following research, implementation and experimental questions:

What are the aspects of actors that plausibly explain interactive (social) behaviour?

To answer this question, we need theories, models and requirements that address the individual as well as the interaction between individuals. The following three sub-questions will elaborate this:

What type of a model can explain interactive (social) behaviour?

Methodological individualism and social constructivism argue that social behaviour should be explained in terms of individuals, their properties and their interrelations in terms of these properties. Multi-Agent Systems as a model and methodology support this view and can serve to explain interactive social behaviour, because MAS is concerned with the study of behaviour, and the modelling of a collection of actors that interact with each other and their environment (Sycara, 1998).

What is required for an actor to exhibit (stable) social behaviour?

For an actor to exhibit social behaviour, it is necessary to create and transfer signs and meaning between actors. Although theories of MAS address coordination, cooperation and negotiation issues, they focus mainly on social structures and often overlook processes that describe how these structures emerge, stabilise and disappear.

Social constructivism and organisational semiotics consider that (stable) social behaviour requires (1) the creation of shared knowledge and the social construction of reality (cf. Berger & Luckmann, 1967), and (2) semiotic resources such as signs/symbols/social constructs that refer to social structures, institutions and habits of action.

What kind of an actor can plausibly handle signs, relations and social constructs?

An actor that handles signs, relations and social constructs needs to have a system that supports representations and mechanisms that manipulate these representations and exhibit intelligent action. A general intelligent actor (system), whatever additional structures and processes it may have, will contain a physical symbol system (Newell & Simon, 1976; Newell, 1980). Therefore, the actor should be equipped with a cognitive architecture. Such a cognitive plausible actor is able to generate and process symbols, create and 'understand' social constructs and build relations with other actors.

Whereas the theoretical research questions explain what theories are applicable to our research in order to implement a Multi-Agent System that is capable of simulating organisational behaviour, the implementation questions give guidance to the development of a software tool and what is necessary to implement such a tool.

How can (cognitive and social) plausible actors be implemented in a Multi-Agent System?

First, the cognitive actor, which is the main component of a MAS, is constructed. A production system is chosen from a selection of different approaches that are possible within AI (cf. Ferber, 1999, p. 125). It is the most elaborate choice that matches the necessary requirements for an actor to be a rational problem solver. A realistic computerised model of a cognitive plausible actor can be created with a cognitive architecture such as SOAR (Lehman et al., 2006; Newell, 1990) or ACT-R (Anderson & Lebiere, 1998). ACT-R was chosen here as the architecture to model the individual actor (see chapter 4 for a discussion). In its most recent release, ACT-R is an architecture of a single actor for which no multi-actor architecture version is yet available. Therefore, the ACT-R architecture is extended and transformed into a new architecture, RBot, that enables ACT-R to become a multi-actor system. Within the newly created MAS (Multi-RBot System or MRS), it is possible to incorporate multiple actors in a task environment.

With the help of this developed MAS, we try to answer the following experimental questions.

Is it possible that social constructs and organisational behaviour can emerge from social interaction?

Is a social construct a coordination mechanism that can influence the behaviour of interacting related actors towards a certain desired behaviour?

The objective of the implementation / experimental questions is to address the mechanisms inside the actor that process and produce social constructs, and are influenced by interactions with other actors. The underlying assumption is that the actor is socially connected to others on a continual basis. The second objective is that these questions are simultaneously design questions. First, the requirements are drawn up. Second, the design or model is constructed. This is followed by an implementation of the system, and fourth, demonstrative experiments show its working. The design questions guide the construction of a model that attempts to bridge the gap between cognitive and social sciences (cf. Sun, 2006a).

The research concludes that the implemented software, called RBot (cf. Roest, 2004), is able to analyse behaviour at different levels and that the software allows for studying the behaviour of actors at the group level (the habituation of behaviour) between individuals, of the individual itself (the rational level) and intra-individual (the functional level: symbolic and sub-symbolic level). Experiments or simulations are run to show that RBot can simulate behaviour

at all these levels. The first experiment is an internal validation showing that RBot is cognitive plausible by comparing its behaviour with that of ACT-R. Secondly, the emergence of social constructs is demonstrated in an experiment composed of a MAS environment whereby actors establish a (tacit) social construct by adapting towards each other's behaviour. The final experiment shows the implementation of an explicit social construct as a representation in the mind of the actor. The social construct is a condition-action pattern implemented at the normative level (meta-cognition) of the actor that influences the processing of productions at the lower level of the actor. In comparison to second experiment, the last experiment shows that a social construct enforces a faster stabilisation and a more predictable way of controlling the behaviour of the overall system (the group).

The conclusion is that this research contributes to a better understanding of how individual behaviour has impact on social behaviour and vice versa. The concept of social construct, because of its different meanings and presence at different levels, connects the social level with the individual level and therefore is a serious attempt to bridge the gap between the cognitive and social level (cf. Alexander & Giesen, 1987).

Finally, the work reported in this dissertation is encompasses both theoretical work and software development. The simulation experiments reported in the dissertation are mere demonstrations of the fact that the developed architecture works as intended; these experiments should not be considered as the core of the work done (as opposed to research consisting of experiments done with pre-existing software packages). The realised software architecture[†] is a tool that enables multi-actor simulations based on intelligent socially aware actors, and is thus a contribution to the community of people engaged in multi-actor simulations.

[†]http://sourceforge.net/projects/act-rbot

Samenvatting

Multi-Agent Systemen (MAS) is een onderzoeksveld dat zich bezig houdt met ontwerp, analyse en implementatie van computationele beslissing en simulatie systemen—een verzameling van actors die met elkaar interacteren—om beter te kunnen begrijpen hoe mensen samenwerken en op welke manier zij hun acties coördineren om problemen op te lossen. MAS begon zich te ontwikkelen gedurende de jaren negentig en propageerde zich op het gebied van cognitie wetenschappen en sociale wetenschappen. Meer specifiek, MAS wordt toegepast in cognitief modelleren en modelleren van multi-actor interactie zoals in bijvoorbeeld sociale simulaties (Conte & Castelfranchi, 1995b; Troitzsch, 1997). Echter, gebleken is dat er niet voldoende interactie is tussen cognitieve en sociale wetenschappen. Wetenschappelijke gereedschappen / methodologie en methodes zijn onvoldoende ontwikkeld (Sun, 2006b). In ons onderzoek brengen we dit probleem naar voren en doen we een poging om beide velden dichter tot elkaar te brengen.

Het doel van het onderzoek is het ontwikkelen en implementeren van een cognitief computationeel sociaal simulatie model gebaseerd op een selectie van sociale en cognitieve theorieën in een poging te voldoen aan de vraag naar een complex cognitief-sociaal model. De motivatie voor ons onderzoek is tweeledig: (1) het ontsluieren van de componenten / onderdelen van MAS en de sociaal cognitieve actor (cf. Conte & Castelfranchi, 1995b) gebaseerd op theoretische beschouwingen met de reden om (2) een simulatie model te construeren dat op een plausibele manier interactief sociaal gedrag kan verklaren van actoren /mensen in een fysiek en sociaal gesitueerde omgeving (cf. Gilbert & Troitzsch, 1999). Het doel is om daarmee gedrag van individuen (microniveau) die een groep of organisatie vormen te relateren aan gedrag en prestatie van een groep of organisatie als geheel (macroniveau) (Alexander & Giesen, 1987; Van den Broek, 2001).

Sociaal gedrag (en gedrag van organisaties) is een uitkomst van interacties tussen individuen. Deze interacties kunnen leiden tot (tijdelijke) stabiele gedragspatronen. Een organisatie kan gezien worden als een groep van mensen met een verzameling gedragspatronen (gewoontes, gebruiken en acties) die tot doel hebben samenwerking en coördinatie van werk mogelijk te maken. Een organisatie is niet een fysiek tastbaar object, zoals een appel of een toetsenbord. De

observatie, het vaststellen en voortbestaan van een organisatie zijn afhankelijk van het bestaan van menselijke gedragingen (gewoontes, gebruiken) en tekens die door mensen worden geproduceerd; het is een product van interactief sociaal gedrag (Helmhout et al., 2004). Een organisatie is vertegenwoordigd als representaties (in de gedachten van actoren) die structuur geven aan de interacties tussen mensen onderling en daarmee een groep van mensen vaststellen die gezien worden als leden van een organisatie (Van Heusden & Jorna, 2001).

De interactie en relatie tussen organisaties, actoren en mentale representaties in de hoofden van actoren is evident; het (gecontroleerd) organiseren is onmogelijk zonder coördinatie en gebruik van representaties tussen actoren onderling. In onze modellen maken we gebruik van het concept sociale constructen: sociale constructen kunnen gezien worden als representaties van samenwerking en coördinatie, gebaseerd op gebruiken (gewoontes) en wederzijdse toezeggingen die vaak uitgedrukt worden in teken structuren zoals overeenkomsten, contracten, plannen, enzovoort (Liu, 2000; Gazendam, 2003, p. 205).

In ons onderzoek passen we een gecombineerde (gemixte) analyse toe (Sun, 2006b) bestaande uit de volgende beschrijvingsniveaus:

1. Het *sociale beschrijvingsniveau* beschrijft sociaal gedrag dat veroorzaakt en beïnvloed wordt door gedrag van individuen. Hieronder valt bijvoorbeeld onderhandelen, het bereiken van overeenstemming, sociale normen en wetgeving.

2. Het *semiotische beschrijvingsniveau* beschrijft het gebruik van taal en tekens in communicatie en interactie welke noodzakelijk zijn om overeenstemming te krijgen over sociale constructen (bv. plannen of contracten) (Gazendam, 2004; Helmhout et al., 2004, 2005b).

3. Het *intentionele beschrijvingsniveau* schrijft kennis (beliefs), wensen en intenties toe aan actoren. Intenties van anderen, afgeleid van kennis over andermans kennis en wensen, maken het mogelijk acties van anderen te onderzoeken, begrijpen en te voorspellen (Dennett, 1987).

4. Het *functionele beschrijvingsniveau* beschrijft leer / cognitieve mechanismen van de actor en is verankerd in empirisch gevalideerde theorie (cf. Anderson & Lebiere, 1998).

5. Het *fysieke/fysiologische beschrijvingsniveau* is een beschrijvingsniveau op het vlak van fysica, biologie en scheikunde. Het voorspelt of verklaart gedrag in termen van natuurkundige wetten of fysiologische eigenschappen (Dennett, 1987).

Het sociale construct (centraal in dit proefschrift) heeft verschillende betekenissen en is aanwezig in / op al deze niveaus. Ten eerste, een sociaal construct is het resultaat van interactie tussen actoren (sociaal niveau). Actoren maken een gewoonte van hun acties en creëren op die manier een habitat van sociale constructen. Teken productie levert codes of tekens (semiotisch niveau) op waaraan actoren (gedeelde) betekenis geven. Deze gedeelde betekenis of kennis wordt normatief op het moment dat deze het gedrag van leden in een gemeenschap

beïnvloedt. Normen kunnen bijvoorbeeld gedrag aansturen zoals het respect hebben voor oudere mensen. Om betekenis te kunnen begrijpen en te veranderen hebben actoren een betekenis systeem nodig (Eco, 1976). Een dergelijk betekenis systeem moet gegrond zijn in de actor hetgeen mogelijk is met behulp van een cognitieve architectuur (het functionele / cognitieve niveau); een cognitieve architectuur stelt een actor in staat om tekens te verwerken en te produceren.

Het sociale construct of een afgeleide van het sociale construct (teken) moet aanwezig zijn op het cognitieve, sociale en semiotische niveau. In de gemeenschap is het sociale construct aanwezig als een teken (structuur). Deze tekens of sociale constructen worden uitgewisseld met behulp van communicatie en rituelen. De uitwisseling is mogelijk door middel van een medium die het transport van sociale constructen in de vorm van berichten, documenten of gebeurtenissen / acties toestaat. Communicatie tussen actoren en het versterken van sociale constructen (veroorzaakt door het frequente gebruik van sociale constructen) zal leiden tot stabiele gedragspatronen en gedeelde normatieve kennis. Behalve dat heeft de actor een fysiek symbool systeem (physical symbol system) nodig (Newell, 1980) die het mogelijk maakt voor de actor representaties op te slaan, tekens/symbolen/sociale constructen te verwerken en produceren, en intelligente acties uit te voeren.

Wij zijn geïnteresseerd in de interactie tussen de verschillende niveaus. In ons onderzoek proberen we niet alleen de interactie te beschrijven maar ook deze interactie te simuleren met behulp van een Multi-Agent Systeem. De combinatie van theorie en MAS heeft geleid tot de volgende onderzoeksvragen, implementatie dan wel experimentele vragen:

> **Wat zijn de aspecten van actoren die interactief sociaal gedrag kunnen verklaren?**

Om deze vraag te kunnen beantwoorden moeten we gebruik maken van theorieën en modellen die zowel het individu als de interactie tussen individuen beschrijven en verklaren. De volgende drie subvragen trachten dit te verduidelijken:

> **Wat voor type model kan interactief (sociaal) gedrag beschrijven en verklaren?**

Methodologisch individualisme en sociaal constructivisme beargumenteren dat sociaal gedrag verklaard zou moeten worden in termen van individuen, hun eigenschappen en eigenschappen van relaties tussen individuen onderling. Een Multi-Agent Systeem als model en methodologie ondersteunt deze beschouwing en kan dienen als hulpmiddel voor het beschrijven en verklaren van interactief sociaal gedrag mede doordat MAS zich toelegt op de studie van gedrag en de modellering van een verzameling van actoren die interacteren met elkaar en hun omgeving (Sycara, 1998).

> **Wat is noodzakelijk voor een actor om (stabiel) sociaal gedrag uit te oefenen?**

Om stabiel gedrag te vertonen is het voor een actor noodzakelijk om tekens en betekenis te creëren en uit te wisselen met andere actoren. Alhoewel theorieën van MAS coördinatie, samenwerking en onderhandelingskwesties analyseren, legt MAS de nadruk op sociale structuren en concentreert zich minder op de processen die beschrijven hoe deze structuren ontstaan. Boven dat veronderstelt sociaal constructivisme en semiotiek (van organisaties) dat (stabiel) sociaal gedrag mogelijk is door middel van (1) de creatie van gedeelde kennis en de sociale constructie van de realiteit (cf. Berger & Luckmann, 1967) en (2) semiotische bronnen zoals tekens/symbolen/sociale constructen die refereren naar sociale constructen, instituties en gedragspatronen (rituelen, gebruiken).

Wat voor soort actor kan op een plausibele manier gebruik maken van / omgaan met tekens, relaties en sociale constructen?

Een actor die gebruik maakt van tekens, relaties en sociale constructen moet een systeem hebben dat representaties en mechanismen kan bevatten; mechanismen die representatie manipuleren en het mogelijk maken voor de actor intelligente acties uit te voeren. Sterker nog, een algemeen intelligente actor (systeem) heeft een fysiek symbolen systeem nodig (physical symbol system) (Newell & Simon, 1976; Newell, 1980). Om die reden moet een intelligente actor uitgerust zijn met een cognitieve architectuur. Een dergelijke cognitief plausibele actor is in staat om tekens te genereren en te verwerken, sociale constructen te creëren en te begrijpen, en relaties op te bouwen met andere actoren.

De theoretische onderzoeksvragen verklaren wat voor theorieën geschikt zijn voor ons onderzoek en maakt het mogelijk om een Multi-Agent Systeem te implementeren welke in staat is sociaal gedrag en gedrag van organisaties te simuleren. De implementatievragen daarentegen geven sturing aan de ontwikkeling van een MAS en wat noodzakelijk is voor de implementatie van een dergelijk systeem.

Op welke manier kunnen (cognitief en sociaal) plausibele actoren geïmplementeerd worden in een Multi-Agent Systeem?

Ten eerste is in dit onderzoek de cognitieve actor vervaardigd; het hoofd onderdeel van ons Multi-Agent Systeem. Een productie systeem is gekozen uit een selectie van verschillende benaderingen die bekend zijn binnen kunstmatige intelligentie (cf. Ferber, 1999, p. 125). Een productie system is de best doordachte keuze die voldoet aan de benodigde eisen voor een rationeel (probleem oplossende) actor. Een realistisch computer model van een cognitief plausibele actor kan gecreëerd worden met behulp van bijvoorbeeld SOAR (Lehman et al., 2006; Newell, 1990) of ACT-R (Anderson & Lebiere, 1998). ACT-R als cognitieve architectuur (zie hoofdstuk 4 voor een discussie) is in deze dissertatie gekozen om de individuele actor te modelleren. ACT-R is een architectuur geschikt voor het modelleren van een enkele actor; tot nu toe is er nog geen versie ontwikkeld van ACT-R waarmee een complete multi-actor architectuur gemodelleerd kan worden. Daarom hebben we de architectuur uitgebreid en omgevormd tot de nieuwe architectuur RBot. Deze nieuwe architectuur maakt het mogelijk voor

ACT-R een multi-actor architectuur te worden. In het nieuwe gecreëerde Multi-Agent Systeem (Multi-RBot Systeem of MRS) is het mogelijk om meerdere actoren te laten interacteren in een taak omgeving.

Met behulp van het ontwikkelde Multi-Agent Systeem (MRS) proberen we de volgende experimentele vragen te beantwoorden.

> **Is het mogelijk dat sociale constructen en (stabiel) gedrag van organisaties kunnen ontstaan uit sociale interactie?**
>
> **Is een sociaal construct een coördinatie mechanisme welke het gedrag van inter-acterende actoren op een bepaalde manier kan beïnvloeden waardoor er een bepaald gewenst gedrag kan ontstaan?**

Het doel van de implementatie / experimentele vragen is de interne mechanismen van de actor te modelleren die sociale constructen verwerken en produceren en beïnvloed worden door interacties met andere actoren. De onderliggende assumptie is dat de actor sociaal gebonden is aan anderen. Het tweede doel is dat deze vragen tegelijkertijd ontwerpvragen zijn. Eerst worden de voorwaarden vastgesteld. Ten tweede wordt het ontwerp of het model geconstrueerd. Vervolgens wordt het systeem geïmplementeerd en als laatste demonstreren experimenten de werking van het systeem. Ontwerpvragen geven richting aan de constructie van het model; een model waarmee een poging wordt gedaan de kloof tussen cognitieve en sociale wetenschappen te overbruggen (cf. Sun, 2006a).

De conclusie van het onderzoek zal laten zien dat de geïmplementeerde software (RBot (cf. Roest, 2004)) het mogelijk maakt gedrag op verschillende niveaus te analyseren; de software geeft mogelijkheden om gedrag van actoren te bestuderen op groepsniveau (gewoontes, rituelen), op het niveau van het individu zelf (rationele niveau) en op het interne niveau van het individu (het functionele niveau: het symbolische en subsymbolische niveau). We hebben experimenten / simulaties uitgevoerd die laten zien dat RBot al deze niveaus kan simuleren.

Het eerste experiment is een interne validatie die laat zien dat RBot cognitief plausibel is door zijn gedrag te vergelijken met dat van ACT-R. Ten tweede, het ontstaan van sociale constructen is aangetoond in een experiment bestaande uit een MAS omgeving waarin actoren een (stilzwijgend) sociaal construct overeenkomen door middel van zichzelf aan te passen aan het gedrag van de ander (de omgeving). Het laatste experiment laat de implementatie van een expliciet sociaal construct zien als representatie in het hoofd van de actor. Het sociale construct is een conditie-actie patroon geïmplementeerd op het normatieve niveau (metacognitie) van de actor. Dit normatieve niveau beïnvloedt het verwerken van producties op het lagere niveau van de actor. Vergeleken met het tweede experiment laat het laatste experiment zien dat een sociaal construct aanstuurt op een snellere stabilisatie en een (meer te voorspellen) manier is voor het aansturen van het gedrag van het gehele systeem (de groep).

De conclusie is dat dit onderzoek bijdraagt aan een beter begrip over hoe individueel gedrag invloed heeft op sociaal gedrag en vice versa. Het concept so-

ciaal construct, mede door zijn verschillende betekenissen en aanwezigheid op verschillende niveaus, verbindt het sociale niveau met het individuele niveau en is daarom een serieuze poging om de kloof tussen het cognitieve en sociale niveau te overbruggen (cf. Alexander & Giesen, 1987).

Tenslotte is het werk dat verricht is in deze dissertatie voornamelijk theoretisch werk en software ontwikkeling. De simulatie experimenten gerapporteerd in de dissertatie zijn louter demonstraties die laten zien dat de ontwikkelde architectuur naar behoren functioneert; deze experimenten moeten niet beschouwd worden als de kern van het werk dat gedaan is (in tegenstelling met onderzoek bestaande uit experimenten gedaan met reeds bestaande software). De gerealiseerde software architectuur[‡] is een gereedschap die het mogelijk maakt multi-actor simulaties te implementeren gebaseerd op intelligente sociaal bewuste actoren. Een dergelijk stuk wetenschappelijk gereedschap is dus een bijdrage aan de gemeenschap van mensen die onderzoek uitoefent met behulp van Multi-Agent Systemen.

[‡]http://sourceforge.net/projects/act-rbot