Introduction to UML-B, UML-B Class Diagrams, UML-B Context Diagrams

May, 2010

Colin Snook University of Southampton

What is UML-B?

A Graphical front-end for Event-B

- Plug-in for Rodin

Not UML …

- Has its own meta-model (abstract syntax)
- Semantics inherited from translation to Event-B

… but has some similarities with UML

- Class Diagrams
- State Machine Diagrams

Translator generates Event-B automatically

What are the benefits?

Visualisation

- Helps understanding
- communication

Faster modelling (allows you to experiment)

- One drawing node = several lines of B
- Extra information inferred from position of elements

  (e.g. if contained in a class or statemachine)

Provides structuring constructs

- Class
- Hierarchical state-machines

Getting Started

Install UML-B using the Rodin update site

- Help – Install, select the main Rodin update site, wait for it to retrieve the categories, select UML-B Modelling Environment under Modelling Extensions.

UML-B Perspective

UML-B New Project Wizard

- Opens a project diagram for you
- Add machines and contexts
- Double click on a machine to open a class diagram
- or on a context to open a context diagram

Look for this icon

UML-B Menu

- Disable automatic translation (Recommended for larger models)

UML-B toolbar button

- Save and translate

Class-oriented problems

In Event-B models, often find a pattern

- Set I (or could be constant or variable)
- Variables $v \in I \rightarrow T$
- Events e (i,..) when $i \in I$, … then $v(i) := x$

I is a set of instances of a class

v is a set of values, one for each instance (a class attribute)

e is a ‘family’ of identical events to assign values to v (a class event)

I.e. trying to represent class-oriented problems

An Event-B model of a class-oriented problem

VARIABLES

allocation , rooms

INVARIANTS

class

attribute (or association)

inv1 : $rooms \in \mathbb{P}(ROOMS)$

inv2 : $allocation \in rooms \leftrightarrow GUESTS$

EVENTS

Initialisation

begin

act1 : $rooms := \varnothing$

act2 : $allocation := \varnothing$

end

class event

Event $Check\_in \mathrel{\widehat{=}}$

any g, r where

grd1 : $g \in GUESTS$

grd2 : $r \in rooms \setminus dom(allocation)$

then

act1 : $allocation(r) := g$

end

constructor

Event $Add\_Room \mathrel{\widehat{=}}$

any r where

grd1 : $r \in ROOMS \setminus rooms$

then

act1 : $rooms := rooms \cup \{r\}$

end

destructor

Event $Remove\_Room \mathrel{\widehat{=}}$

any r where

grd1 : $r \in rooms \setminus dom(allocation)$

then

act1 : $rooms := rooms \setminus \{r\}$

end

Example - modelling with UML-B

In a university degree programme, students are registered on degree courses.
Students must be enrolled to be registered in a course.
Courses can be removed from the degree programme.

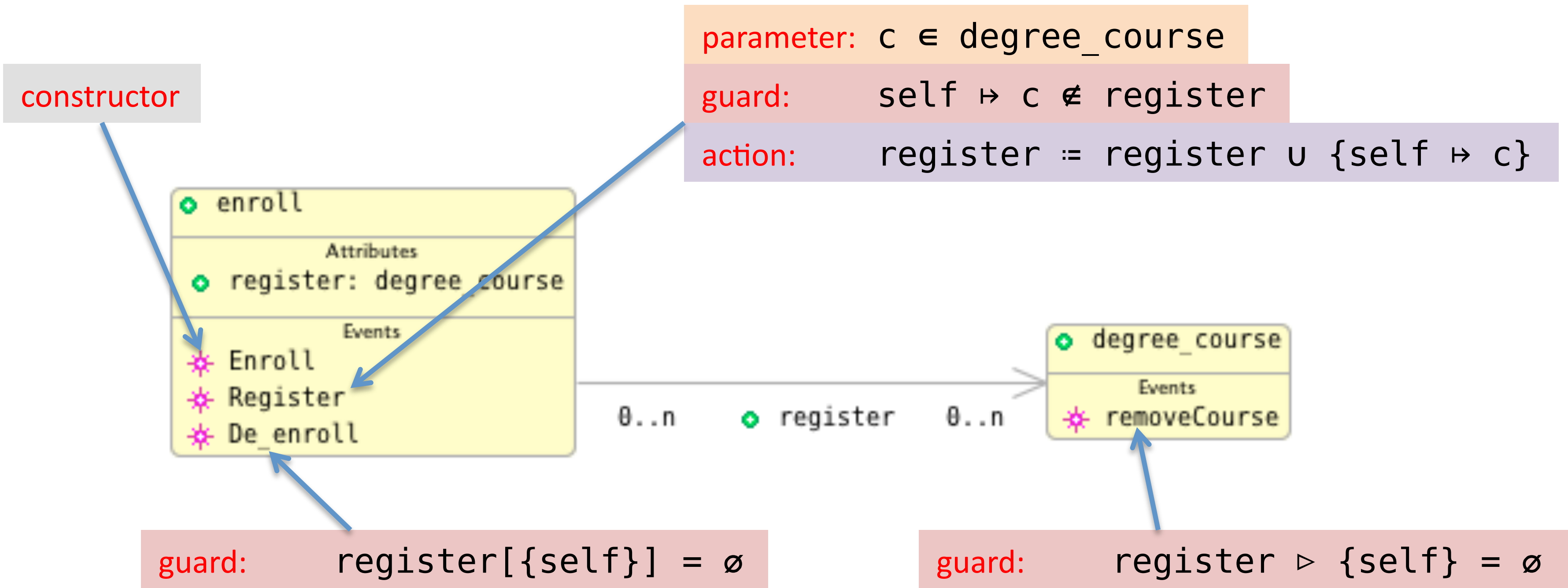
There is no need to consider multiple degree programmes - just assume we are modelling a single degree programme.

Example - modelling with UML-B

In a university degree programme, students are registered on degree courses.
Students must be enrolled to be registered in a course.
Courses can be removed from the degree programme.

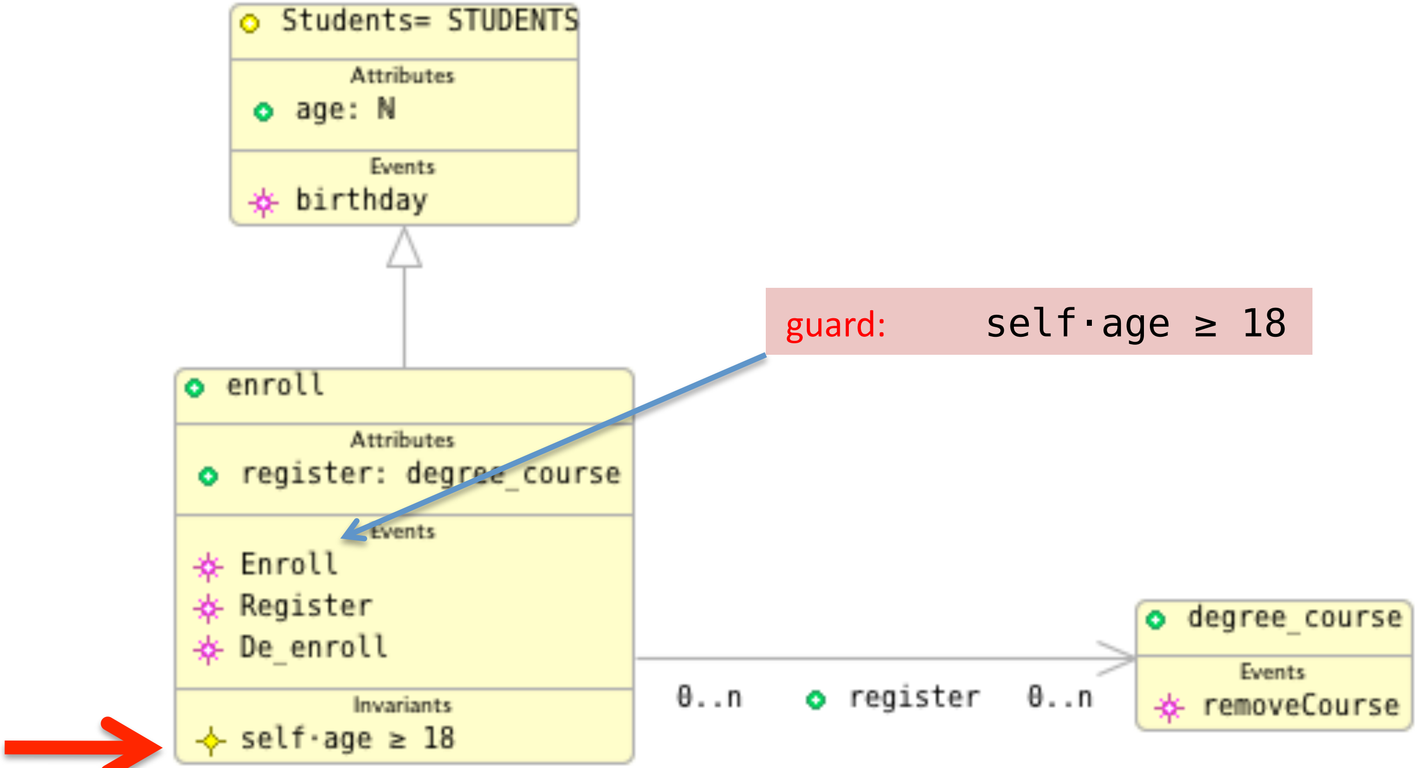
Example - modelling with UML-B

In a university degree programme, students are registered on degree courses.
Students must be enrolled to be registered in a course.
Courses can be removed from the degree programme.



Adding an Invariant

Enrolled students must be 18.



Translation: $\forall self \cdot ((self \in enroll) \Rightarrow (age(self) \geq 18))$

UML-B Class Diagrams – Translation rules (part)

UML-B	Event-B
Class (variable instances) Class (fixed instances) Class (variable inst and has super class) Class (fixed inst and has super class)	Variable $\subseteq$ Set Set Variable $\subseteq$ SuperClass Constant $\subseteq$ SuperClass
Attribute (card 0..n - 1..1) Attribute (card 0..n - 0..1) Attribute (card 0..n - 0..n) Etc. (try other cardinalities in UML-B)	Variable $\in$ Class $\rightarrow$ Type Variable $\in$ Class $\nrightarrow$ Type Variable $\in$ Class $\leftrightarrow$ Type Etc.
Associations	As Attribute but Type is another class
Class Event	Event(self) WHEN self $\in$ Class …
Class Constructor	Event(self) WHEN self $\in$ SET\Class …
Class Invariant	$\forall$self$\cdot$((self$\in$Class)$\Rightarrow$ Class invariant

Example – Event-B produced by UML-B

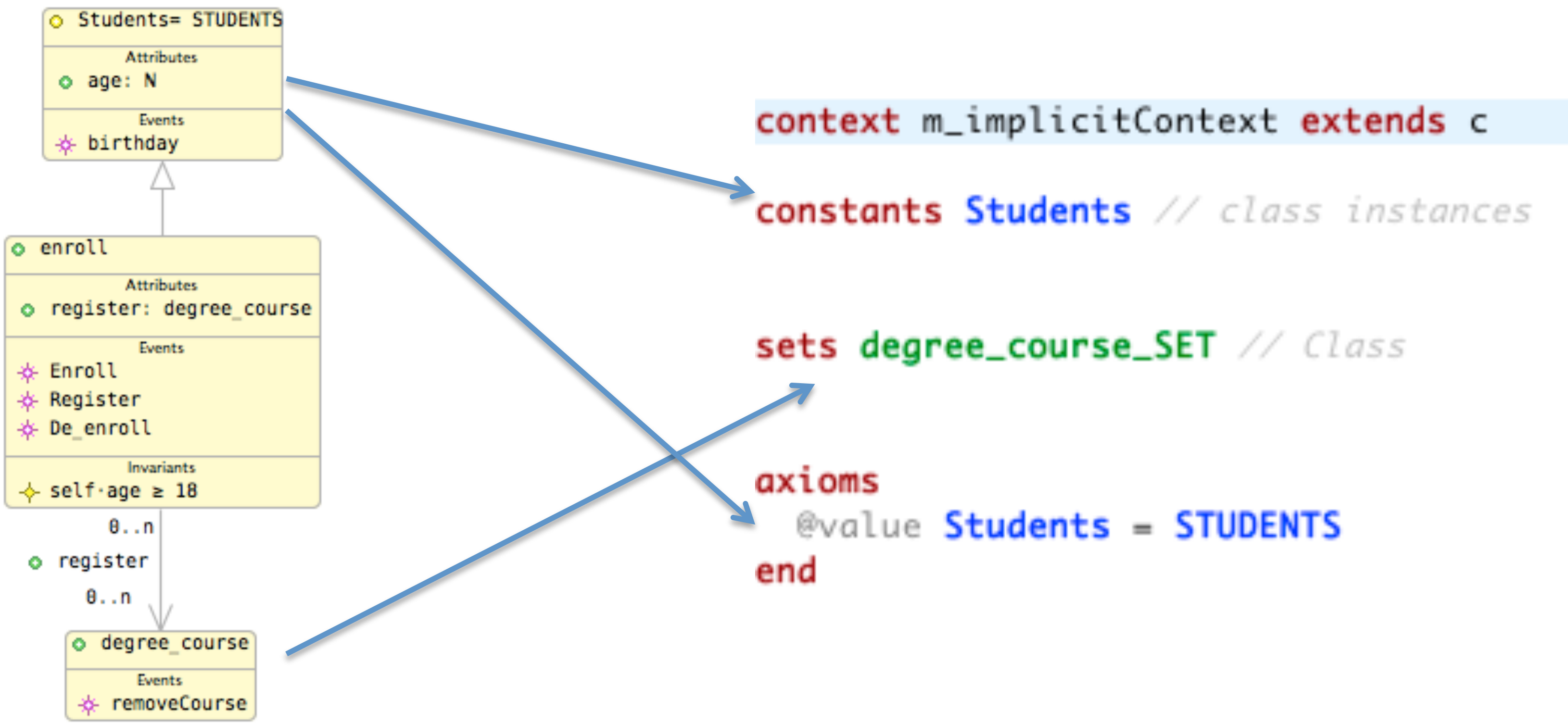
```
machine m sees m_implicitContext

variables enroll // class instances
          degree_course // class instances
          register // attribute of enroll
          age // attribute of Students


invariants
  @type enroll ∈ ℙ (Students)
  @type degree_course ∈ ℙ (degree_course_SET)
  @type register ∈ enroll ↔ degree_course
  @type age ∈ Students → ℕ
  @Invariant1 ∀self·((self∈enroll)⇒(age(self) ≥ 18))

events
  event INITIALISATION
    then
      @init enroll ≔ ∅
      @init degree_course ≔ ∅
      @init register ≔ ∅
      @init age ≔ Students × {0}
  end
```

```
event Enroll
  any self // constructed instance of class enroll

  where
    @type self ∈ Students \ enroll
    @Guard1 age(self) ≥ 18
  then
    @enroll_constructor enroll ≔ enroll ∪ {self}
end

event Register
  any self // contextual instance of class enroll
      c
  where
    @type c ∈ degree_course
    @type self ∈ enroll
    @Guard1 self ↦ c ∉ register
  then
    @Action1 register ≔ register ∪ {self ↦ c}
end
```

The 'Implicit' Context

Each class diagram creates an *implicit* context

- Contains the 'basis' of things on the class diagram
- e.g. a carrier set for the type of class instances



Context Diagrams

How can we model constants that belong to a class?

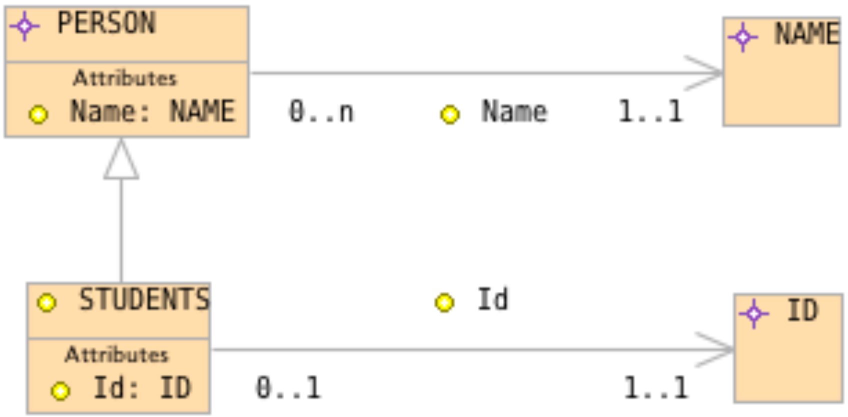
in Event-B our machine would see a Context

with sets, constants, axioms

UML-B takes a similar approach

- Class Diagram (Machine) sees Context Diagram
- Similar to a Class Diagram but translates to sets, constants and axioms
- ClassType instead of Class
- Constant Attributes/Associations represent constants
- Axioms instead of Invariants
- No Events

A Context Diagram and its translation



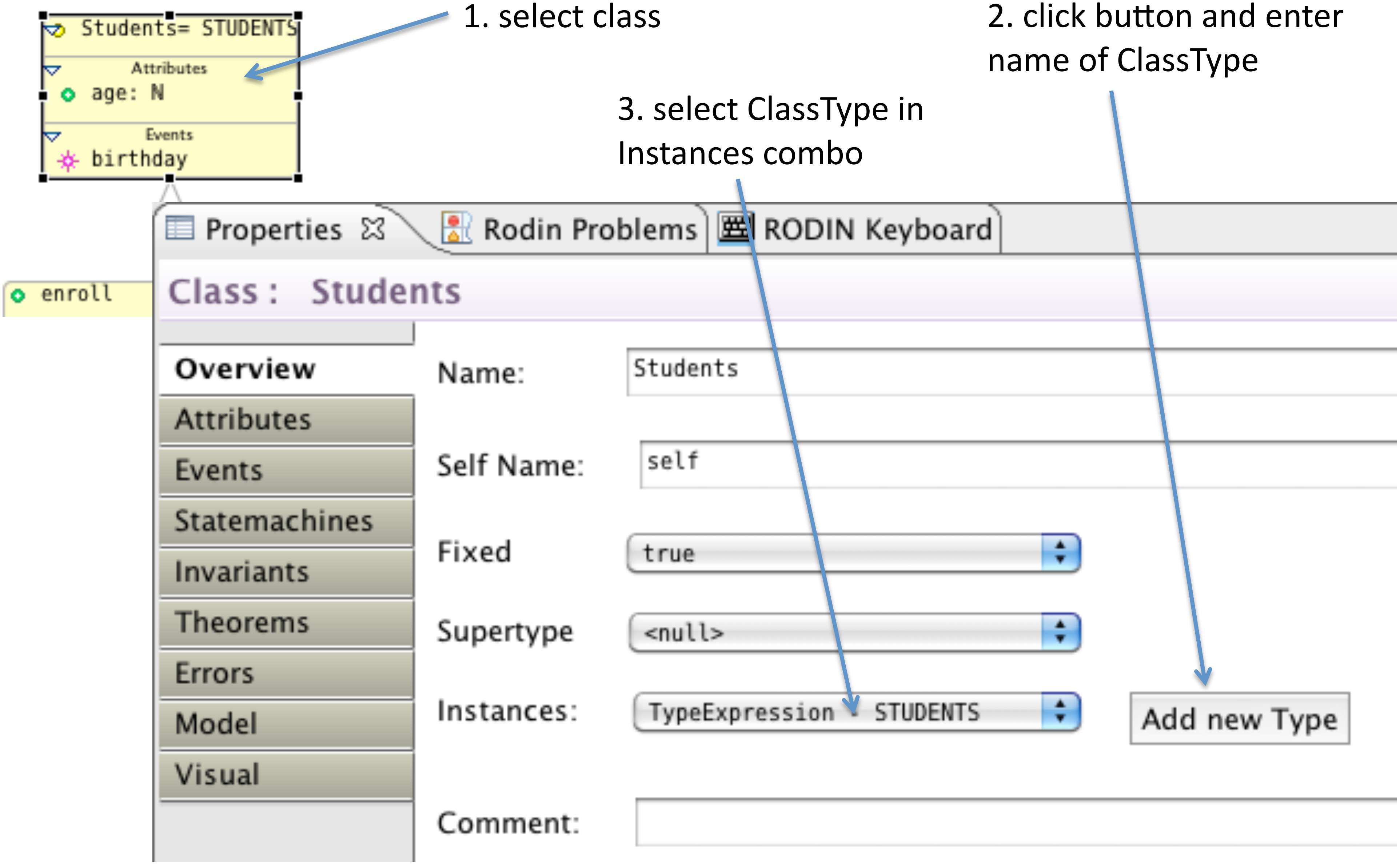

```
context c

constants STUDENTS // classType instances
          Id // attribute of STUDENTS
          Name // attribute of PERSON


sets ID // ClassType
     PERSON // ClassType
     NAME // ClassType


axioms
  @type STUDENTS ∈ ℙ (PERSON)
  @type Id ∈ STUDENTS ↣ ID
  @type Name ∈ PERSON → NAME
end
```

Linking a Class to a ClassType

1. select class

2. click button and enter name of ClassType

3. select ClassType in Instances combo

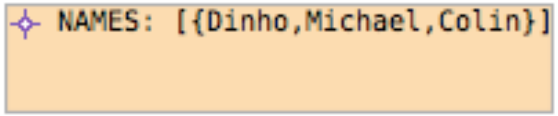


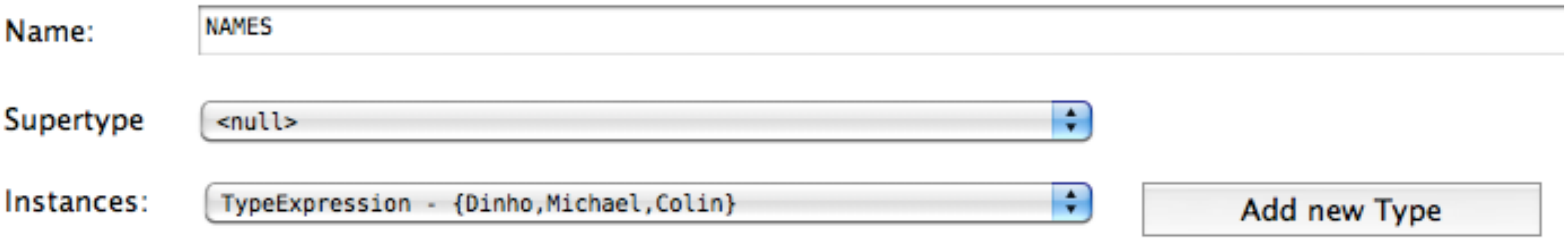
Enumerated Types

For real enumerated types e.g.

signal = {proceed, warning, stop}

also, for restricting types to an example for model checking





Enumerated Types

```
CONTEXT
  Context1

SETS
  NAMES      //   ClassType

CONSTANTS
  Dinho      //   enumeration constant
  Michael      //   enumeration constant
  Colin      //   enumeration constant

AXIOMS
  Dinho.type    :    Dinho ∈ NAMES
  Michael.type    :    Michael ∈ NAMES
  Colin.type    :    Colin ∈ NAMES
  partition enumeration of NAMES    :    partition(NAMES,{Dinho},{Michael},{Colin})

END
```

Summary

Class diagrams for class-oriented modelling

- automatically generates class structures in Event-B

Attribute and association cardinalities

Options for class instances

- variable (constructors and destructors)
- fixed

Automatically generates an ‘implicit context’

Context diagrams for class oriented modelling of sets, constants and enumerated types