Requirements, Traceability and DSLs in Eclipse with the Requirements Interchange Format (RIF/ReqIF)

Andreas Graf Michael Jastram

andreas.graf@itemis.de jastram@cs.uni-duesseldorf.de

Abstract: Requirements engineering (RE) is a crucial aspect in systems development and is the area of ongoing research and process improvement. However, unlike in modelling, there has been no established standard that activities could converge on.

In recent years, the emerging Requirements Interchange Format (RIF/ReqIF) gained more and more visibility in industry, and research projects start to investigate these standards. To avoid redundant efforts in implementing the standard, the VERDE and Deploy projects cooperate to provide a stable common basis for RIF/ReqIF that could be leveraged by other research projects too. In this paper, we present an Eclipse-based extensible implementation of a RIF/ReqIF-based requirements editing platform.

In addition, we are concerned with two related aspects of RE that take advantage of the common platform. First, how can the quality of requirements be improved by replacing or complementing natural language requirements with formal approaches such as domain specific languages or models. Second, how can we establish robust traceability that links requirements and model constructs and other artefacts of the development process. We present two approaches to traceability and two approaches to modelling.

We believe that our research represents a significant contribution to the existing tooling landscape, as it is the first clean-room implementation of the RIF/ReqIF standard. We believe that it will help reduce gaps in often heterogeneous tool chains and inspire new conceptual work and new tools.

1 Introduction

Requirements engineering (RE) is an important aspect of the systems engineering process [Poh07]. In this paper, we explore two aspects of the role of RE in systems engineering. First, we present a framework that allows better integration on the tool level by using the emerging RIF/ReqIF standard [OMG]. Second, we explore integration on the process level by looking at traceability to more formal system representations.

We describe a tool platform for managing natural language requirements that is suited for integration in system development environments. Our aim is to make integration in existing environments easy, to increase the confidence that the quality of the requirements is adequate, and that their relationship to other system elements is understood. We achieve this by providing a robust, extensible software platform for managing requirements. The platform is build on the emerging RIF/ReqIF standard, which facilitates integration with tool chains used in industry.

We demonstrate the extensibility of the platform by presenting two approaches to traceability and two approaches to modelling, implemented on top of the tooling platform. The capturing of requirements that conform to a set of quality criteria is regarded essential for good requirements [Poh07]. A subset of these criteria can be addressed by extending or replacing natural language with more formal notations [GJGZ00]. Formal notations come in a variety of flavours, including mathematical notations (Section 4.1) and domain specific languages (Section 4.3).

On the significance on RIF/ReqIF and our first-clean room implementation of the standard, we draw comparisons to model-driven software development: After the specification of UML, a lot of publications and work concentrated on this standard, paving the way for low-cost and open-source tools. We believe that our open-source clean-room implementation of the standard based on Eclipse can serve as the basis for both innovative conceptual work and new tools.

We introduce two European research projects that address these challenges with tooling based on Eclipse and the requirements standard RIF/ReqIF.

1.1 The Verde and Deploy Research Projects

The research project Verde[1] has the aim of providing a universal tool platform for the verification- and validation-orientated development of embedded systems. The focus is on the integration of the tools which are already in use at the industrial partners. Verde develops new tools and methods in the areas where there are gaps in existing tool-chains and procedures.

Deploy[2] is an European Commission Information and Communication Technologies FP7 project. Its goal is to make major advances in engineering methods for dependable systems through the deployment of formal engineering methods. Deploy uses the Event-B [Abr10] formal method as a basis (Section 4.1), for which tool support in the form of the Rodin platform [ABHV06] exists.

1.2 Related Work

We use natural language requirements (NLRs) in our work. While requirements can be stored in forms other than natural language, it is the most natural way for the customer to express their perception of the model [Ger97]. NLRs can also be processed (semi-) automatically [AG06], which we won't pursue in this paper.

The issue of traceability has been analysed in depth by Gotel et. al. [GF94]. Our research falls into the area of post-requirements specification traceability.

Abrial [Abr06] recognizes the problem of the transition from informal user requirements

[1] http://www.itea-verde.org
[2] http://www.deploy-project.eu/

to a formal specification. He suggests to construct a formal model for the user requirements, but acknowledges that such a model would still require informal requirements to get started. He covers this approach in [Abr10].

For traceability between natural language requirements and formal Event-B models, ProR currently supports the WRSPM reference model [GJGZ00]. This model was attractive, because it deliberately left enough room to be tailored to specific needs, as opposed to more concrete methods like Problem Frames [Jac01] or KAOS [DDMvL97]. It is also more formal and complete than the functional-documentation model [PM95], another well-known approach.

Another approach to requirements used in industry is to "graft" requirements on top of another meta-model. An example is SysML, where requirements are represented with the stereotype ≪*Requirement*≫. However, this has the drawback that not all meta-models support extensions [CM10]. In addition, traceability is limited to artefacts that can be represented within the model.

We are aware of a number of system development environments in the open source that support natural language requirements. These include Topcased [FGC+06] and UniCase [BCHK07]. Also, SysML [Wei07] introduces a requirement diagram. However, the data structures for requirements used by these tools and standards lack the richness of the RIF data model. In fact, only commercial tools seem to offer data models that come close to the richness of the RIF data model. These include IBM Rational DOORS[3] or Visure IRQA[4].

Existing traceability solutions can be found as an integral component of a tool or as external add-ons. Topcased and UniCase both support the former approach, which is also used by some commercial tools.

Some commercial traceability tools assume a heterogeneous tool chain. They attempt to map the meta-models of the tools by providing tool- or model-specific adapters (e.g. Reqtify[5]).

1.3 Structure of this Paper

In Section 2, we provide an overview of the RIF/ReqIF Requirements Interchange Format. Section 3 describes the Eclipse-based tool and application framework that we created. Section 4 describes the integration of other elements of the development process using domain specific languages (DSLs), formal Event-B models and traceability constructs. It shows how the tool platform facilitates the integration. Section 5 concludes and provides an outlook on our future plans.

[3] http://www.ibm.com/software/awdtools/doors/
[4] http://www.visuresolutions.com/irqa-requirements-tool
[5] http://www.geensoft.com/en/article/reqtify/

2 The RIF/ReqIF Requirements Interchange Format

RIF/ReqIF [OMG] is an emerging standard for requirements exchange, driven by the German automotive industry. It consists of a data model and an XML-based format for persistence.

2.1 History of the RIF/ReqIF Standard

RIF was created in 2004 by the "Herstellerinitiative Software", a body of the German automotive industry that oversees vendor-independent collaboration. Within a few years, it evolved to the current version 1.2. The format gained traction in the industry, and a number of commercial tools support it. For instance, Rational DOORS claims to support it, and we are aware of the commercial synchronization tools Atego Exerpt[6] and enso ReqIF Server[7]. We are aware of at least one German car manufacturer where the use of RIF plays a strategic role.

In 2010, the Object Management Group (OMG) took over the standardization process and released the ReqIF 1.0 RFC (Request For Comments). The name was changed to prevent confusion with the Rule Interchange Format, another OMG standard, while the version number was reset to 1.0.

Our tool environment is currently based on RIF 1.2, support for ReqIF 1.0 is planned.

2.2 The Content and Structure of a ReqIF Model

In general terms, a ReqIF model contains attributed requirements that are connected with attributed links. The requirements can be arbitrarily grouped into document-like constructs. In the following, we point out a few key model features:

A *SpecObject* represents a requirement. A SpecObject has a number of *AttributeValues*, which hold the actual content of the SpecObject. SpecObjects are organized in *Specifications*, which are hierarchical structures holding *SpecHierarchy* elements. Each SpecHierarchy refers to exactly one SpecObject. This way, the same SpecObject can be referenced from various SpecHierarchies.

ReqIF contains a sophisticated data model for *Datatypes*, support for permission management, facilities for grouping data and hooks for tool extensions. The details can be found in the ReqIF specification [OMG].

[6] http://www.atego.com/products/atego-exerpt-synchronizer/
[7] http://reqif.de/

3 Tool Implementation

The Verde-Project produced an EMF[8]-based implementation of the RIF 1.2 data model, which represents the foundation for both the Verde and the Deploy projects. The two projects provide their own GUI that is adapted to the corresponding approach. Due to the reliance on EMF, both tool solutions can be installed into any Eclipse-based system for deployment.

The Verde-project will publish its tool as part of the Yakindu project[9], which is already available as open source project.

3.1 ProR Requirements Engineering Platform

The tool from the Deploy project is called ProR and is available for download[10]. It is a full-featured editor for RIF-Documents.

The GUI of ProR is shown in Figure 1. The left pane shows projects that can be expanded to show their content. The view in the middle shows the RIF Specifications, which can be customized to show only selected Attributes in a table view. For the selected SpecObject (the selected row), all Attributes are shown in the Properties View on the bottom. Values can be edited in the Property View or directly in the main view. On the right side is an outline view that allows an alternative navigation of the model.

Figure 1 also shows how an integration with other tools may look like. Here we see the Rodin integration. Variables from the formal model are recognized and rendered in colour in the main view. (see also Section 4.1).

3.2 ProR in the Field

We are currently evaluating ProR in the Deploy project, where we model a Cruise Control system (for cars). The system is described by 46 requirements and modelled in Event-B in three refinement levels. As this project is not yet concluded, we cannot report traceability statistics yet.

Further field studies are planned.

[8] Eclipse Modeling Framework, http://www.eclipse.org/emf/
[9] http://www.yakindu.com/
[10] http://pror.org

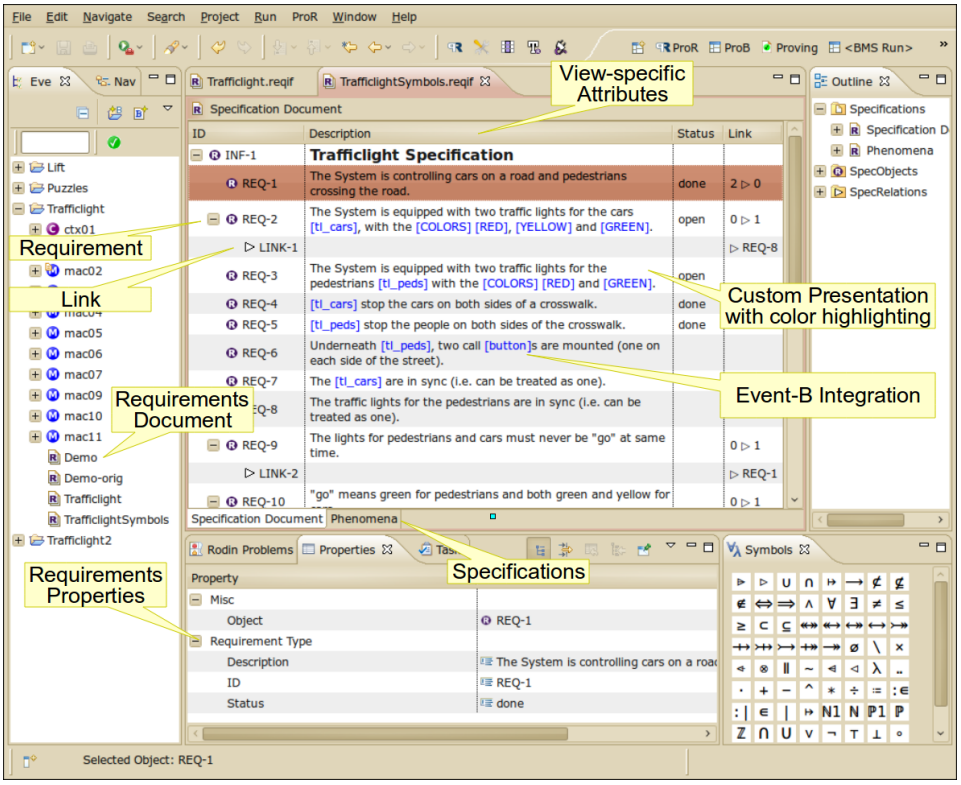

Figure 1: The ProR GUI (here shown running inside the Rodin Platform)

4 Traceability and Integration with Models

In this Section, we show the integration of requirements with two modelling approaches, Event-B (Section 4.1) and DSLs (Section 4.3). We also show two approaches to traceability, using RIF SpecRelations (Section 4.1) and external Tracepoints (Section 4.2).

Traditionally requirements are expressed in natural language. However, natural language suffers from the disadvantage of not being precise enough and impedes (automated) analysis. Accordingly, requirements engineering recommends to find clearer wording by means of formal notations [CM99]. The engineer can directly write down the requirements in textual formal language or replenish them with other models.

To assure that the requirements are fully implemented in the system, it is necessary to trace them during the whole development process [GF94]. The effect of a requested change in any of the artefacts can be assessed by following the traces up and down the process.

4.1 Traceability to Event-B Models via WRSPM

In the Deploy project, we demonstrate our ideas using Event-B, a formalism and method for discrete systems modelling. Event-B is a state-based modelling method. The choice of Event-B over similar methods [BS03, Jon90] is mostly motivated by the built-in formal refinement support and the availability of the Rodin tool [ABHV06] for experimentation with our approach.

Event-B models are characterized by *proof obligations*. Proof obligations serve to verify properties of the model. To a large degree, such properties originate in requirements that the model is intended to realize. Eventually, we expect that by verifying the formal model we have also established that the requirements to which they correspond are satisfied. We developed an approach to traceability between natural language requirements and Event-B models based on WRSPM [JHLR10]. WRSPM is a reference model for applying formal methods to the development of user requirements and their reduction to a behavioural system specification [GJGZ00]. WRSPM distinguishes between artefacts and phenomena. Phenomena describe the state space (and state transitions) of the domain and system, while artefacts represent constraints on the state space and the state transitions. The artefacts are broadly classified into groups that pertain mostly to the system versus those that pertain mostly to the environment.

Our goal is to establish requirements traceability from natural language requirements to an Event-B formal model, using WRSPM to provide structure to both the requirements and the model. To achieve this, we introduce a "realize" relationship that we refine into different types of traces, as well as criteria to verify that the traces have been established correctly. This allows us to "justify" that the formal model realizes the requirements.

We created tool support by using the *SpecRelations* construct in RIF. ProR runs as a feature directly inside Rodin. The integration code is contained in its own plugin, which keeps ProR and Rodin independent.

Via this plugin, ProR supports highlighting of formal model elements directly in the requirements text, as can be seen in Figure 1. It also support the creation of SpecRelations between formal model elements and requirements in ProR. Formal Event-B elements have a corresponding proxy SpecObject in the RIF model that is automatically synchronized with the Event-B model. The integration is currently manual via drag and drop. We plan semi-automatic creation and maintenance of the SpecRelations in the future.

4.2 Tracepoint Approach to Traceability

The general concept of traceability in VERDE led to the decision to implement a traceability that is independent of the types of artefacts that are involved. Since Eclipse-based models are usually built on the Eclipse meta-model Ecore/EMF [SBPM09], VERDE implements a generic solution for the traceability of EMF-based elements (see Figure 2). The core data structure is a mapping table with three elements: source element, target element and arbitrary additional information. The elements are identified by a data structure, the

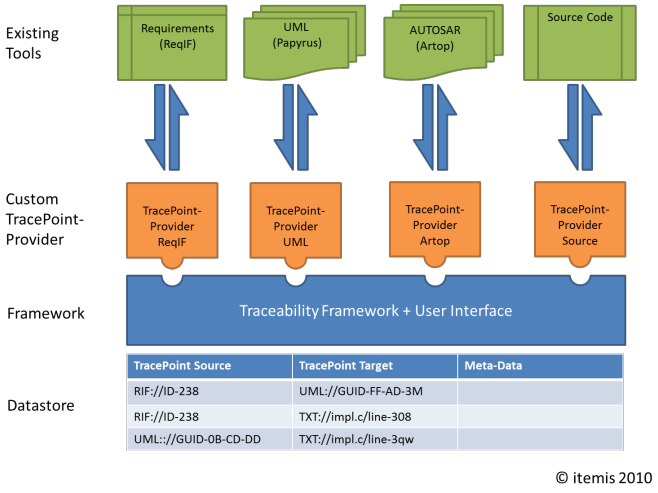

Figure 2: Tracepoint traceability

so called *Tracepoint*. The inner structure of a Tracepoint depends on the structure of the meta-model that is being traced, but is hidden from the traceability infrastructure. For instance, a UML model element may be identified by a unique ID. A C source file by contrast could be identified by file path and name. The basic implementation supplies an "assignment window" that integrates into existing editors (like UML, C-IDE). This allows the user to navigate and edit the traceability relationship.

While we just demonstrated traceability "downstream" (i.e. towards the implementation), it could also be used "upstream" (i.e. towards goals). For instance, project management could provide traceability to milestones that are part of the project plan.

4.3 Integration of Domain-Specific Languages

The possibility to specify requirements with textual domain specific languages (DSLs) and to trace these to development artefacts is one of the foundations of the VERDE project. A textual DSL is a machine-processable language that is designed to express concepts of a specific domain. The concepts and notations used correspond to the way of thinking of the stakeholder concerned with these aspects while still being a formal notation. In the Verde requirements editor, the open-source tool Xtext [EV06] is used. The editor for the DSLs integrates itself directly into the requirements tool. The introduction of Xtext allows any project to define their own grammar and modelling. Project partners can design and evaluate new formal notations for the specification of requirements, in particular non-functional requirements such as memory usage and timing.

4.4 The Role of Tool Support

The approaches presented in this Section could have been realized without a common requirements platform. However, using a common standard and platform makes interoperability much more powerful. Due to the common standard, other tools can directly query and modify the RIF data model, which is not tool-dependent. Further, interoperability with industry tools is possible (as long as they support RIF).

On the platform level, Features like DSL-support or Tracepoint-traceability can be implemented seamlessly via plugins, thereby getting the basic features for requirements management "for free". Multiple plugins can be used without interfering with each other.

5 Conclusion and Future Plans

We believe that our research represents a significant contribution to the existing tool landscape. ProR is the first open source requirements platform supporting RIF. Until now, there were few options between commercial tools that support rich data structures and open source efforts that either use non-standard data models or simple data models like SysML-requirements. The choice of Eclipse and EMF as the platform for ProR makes the integration into other Eclipse-based offerings easy.

The integration is further simplified by offering two orthogonal frameworks for traceability, either using the RIF data structures or an external data model for traceability.

There are many aspects of our research that we are currently exploring. The Verde and Deploy projects will continue until 2012, and we plan to work on RIF/ReqIF compatibility, advanced modelling support, syntactic and semantic analysis, integration with other platforms and collaboration features.

The commercial partners of Deploy and Verde show strong interest in ProR, so we expect that work will continue beyond Deploy and Verde, either in the form of another research project or as a commercial effort.

References

[ABHV06] J.-R. Abrial, M. Butler, S. Hallerstede, and L. Voisin. An open extensible tool environment for Event-B. In *International Conference on Formal Engineering Methods (ICFEM)*, LNCS, New York, NY, 2006. Springer-Verlag.

[Abr06] J.-R. Abrial. Formal Methods in Industry: Achievements, Problems, Future. In *Proc. of the 28th int. conf. on Software engineering*, pages 761–768, 2006.

[Abr10] J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 1 edition, June 2010.

[AG06] V. Ambriola and V. Gervasi. On the Systematic Analysis of Natural Language Requirements with C IRCE. *Automated Software Engineering*, 13(1), 2006.

[BCHK07] B. Bruegge, O. Creighton, J. Helming, and M. Kögel. Unicase-an ecosystem for unified software engineering research tools. In *Third IEEE International Conference on Global Software Engineering, ICGSE*, volume 2008, 2007.

[BS03] E. Börger and R. Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer-Verlag, 2003.

[CM99] R. G. Clark and A.M.D. Moreira. Formal Specifications of User Requirements. *Automated Software Engineering*, 6(3):217–232, 1999.

[CM10] O. Casse and M. R. Monteiro. A ReqIF/SysML profile example - Requirements exchange and roundtrip. In *ERTS2 2010*, volume 32, 2010.

[DDMvL97] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde. GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering. In *Proc. of the 19th int. conf. on Software engineering*, pages 612–613. ACM, 1997.

[EV06] S. Efftinge and M. Völter. oAW xText: A framework for textual DSLs. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32, 2006.

[FGC+06] P. Farail, P. Gaufillet, A. Canals, C. Le Camus, D. Sciamma, P. Michel, X. Crégut, and M. Pantel. The TOPCASED project: a toolkit in open source for critical aeronautic systems design. *Embedded Real Time Software (ERTS)*, 2006.

[Ger97] V.A.V. Gervasi. Processing Natural Language Requirements. In *Automated software engineering: 12th IEEE international conference: proceedings, November 1-5, 1997, Incline Village, Nevada, USA*, page 36, 1997.

[GF94] O. Gotel and A. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of the First International Conference on Requirements Engineering*, page 94–101, 1994.

[GJGZ00] C.A. Gunter, M. Jackson, E.L. Gunter, and Pamela Zave. A Reference Model for Requirements and Specifications. *IEEE Software*, 17:37–43, 2000.

[Jac01] M. Jackson. *Problem Frames: Analysing and Structuring Software Development Problems*. Addison-Wesley/ACM Press, 2001.

[JHLR10] M. Jastram, S. Hallerstede, M. Leuschel, and A.G. Russo Jr. An Approach of Requirements Tracing in Formal Refinement. In *VSTTE*. Springer, 2010.

[Jon90] C.B. Jones. *Systematic Software Development Using VDM*. Prentice Hall, 1990.

[OMG] OMG. Requirements Interchange Format (ReqIF) 1.0 Beta 1.

[PM95] D.L. Parnas and J. Madey. Functional documents for computer systems. *Science of Computer programming*, 25(1):41–61, 1995.

[Poh07] K. Pohl. *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Dpunkt Verlag GmbH, 1 edition, February 2007.

[SBPM09] D. Steinberg, F. Budinsky, M. Peternostro, and E. Merks. *EMF Eclipse Modeling Framework*. Addison-Wesley, 2 edition, 2009.

[Wei07] T. Weilkiens. *Systems engineering with SysML/UML: modeling, analysis, design*. Morgan Kaufmann, 2007.