

ProR, an Open Source Platform for Requirements Engineering based on RIF

Michael Jastram (Heinrich-Heine University Düsseldorf)

Presented at SEISCONF 2010

Abstract

Proper tool support in requirements engineering is important. A number of proprietary solutions are available in the market place, but few open solutions, and none that are widespread at this time. In this paper, we introduce ProR, an open platform for requirements engineering.

The project was initiated by the EU FP7 Project “Deploy”, which has the goal to make major advances in engineering methods for dependable systems through the deployment of formal engineering methods. Rather than building a project-specific solution, we decided to develop an extensible, general purpose platform.

We based the tool’s data model on the Requirements Interchange Format (RIF / ReqIF), a standard driven by the automotive industry. This gives us interoperability with a number of existing tools. The tool is based on the Eclipse Platform and uses the Eclipse Modeling Framework (EMF). We managed to establish a cooperation with the ITEA-Project “Verde”, which supplied the implementation of the RIF data model.

In this paper, we will introduce the tool, describe its architecture, and present a small case study. ProR is still under development, and we welcome contributors and are very interested in potential users to validate our approach. Please visit www.pror.org for more information, and to download the current development version.

1 Overview

In this report, we introduce the ProR platform for requirements engineering (RE), an open source effort. Our goal is to raise awareness of the tool, validate our plans against the needs of the community, and to recruit users, advocates and contributors.

Figure 1 shows the ProR GUI and identifies some major features. Please note that development on ProR only started recently, and it is far from feature complete. Please see Section 6 for the current state of development.

1.1 Problem Statement and Background

Tool support for RE is an important aspect in systems development, particularly in industrial settings. A number of proprietary tool solutions are available, but few open solutions, and none that are currently widespread.

The vision of ProR is to provide reliable traceability between natural language requirements and formal models. Interoperability with existing industrial processes is also a requirement. This approach will allow interested parties to customize ProR for their needs, without having to develop yet another RE tool.

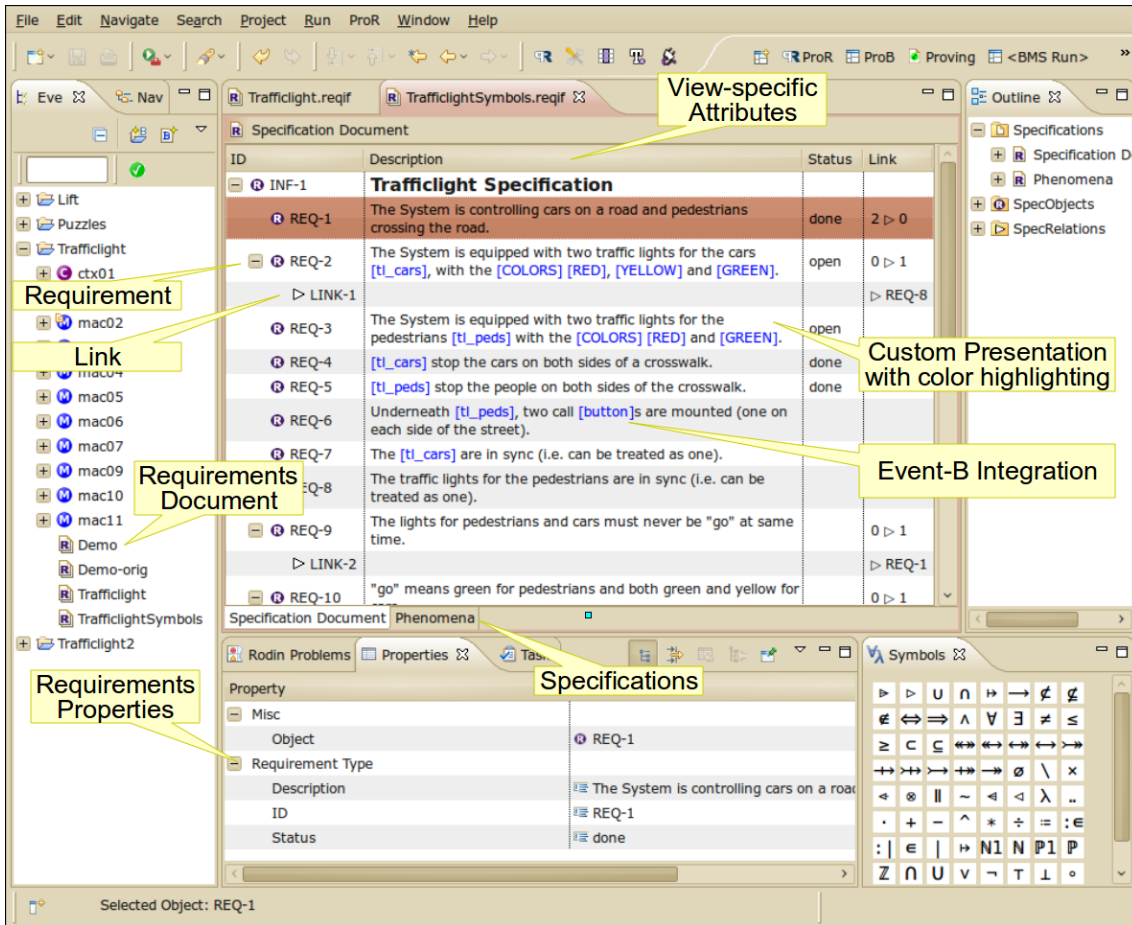


Figure 1: The ProR GUI, running inside the Rodin Platform. See Section 3 for details.

1.2 Deploy and Rodin

The development of ProR was initiated by the EU FP7 Project “Deploy” [Dep], which has the goal to make major advances in engineering methods for dependable systems through the deployment of formal engineering methods.

Part of Deploy is the ongoing work on the Rodin Platform [CJO⁺05], an Eclipse-based IDE for Event-B [Abr10] that provides effective support for refinement and mathematical proof. The platform is open source, contributes to the Eclipse framework and is further extensible with plugins.

Requirements engineering and especially requirements traceability is an integral part of Deploy [Jas09]. A number of methods have been explored: Problem Frames [Jac01], Parnas Four-Variable-Method [FBWJ02], KAOS [DDML97] and WRSPM [GJGZ00], amongst others.

Rather than building a project-specific solution, we decided to develop ProR as an extensible, general purpose platform. This will make it easy to quickly realize tool support for the various RE-methods that we explored.

Our WRSPM-based an approach to traceability between natural language requirements and formal models [JHLR10] reached a point where tool support would be useful. We will use ProR as a generic platform to implement specific support for traceability from natural language requirements to Event-B formal models.

2 Technologies

In this section, we will briefly introduce the various technologies that ProR uses.

2.1 Eclipse

Eclipse [Ecl] is a platform for general purpose applications with an extensible plug-in system. It is mainly known as an integrated development environment (IDE) for Java development, although the Java IDE is just one specialized application of the platform. Eclipse employs plug-ins in order to provide all of its functionality on top of the runtime system which is based on Equinox, an OSGi standard compliant implementation.

The Eclipse platform provides facilities for workspace management, GUI building, a help system, team support and more. These components consist of plugins. Plugins may provide extension points, to which other plugins may connect via extensions. A typical Eclipse installation contains hundreds of extensions.

ProR can run as a stand-alone Eclipse application, or it can be installed into any existing Eclipse installation, including Rodin.

2.2 Eclipse Modeling Framework (EMF)

The Eclipse Modeling Framework [EMF] is a modeling and code generation facility. EMF provides tools and runtime support to produce Java code for the model and adapter classes that enable viewing and command-based editing of the model.

EMF is attractive for ProR for a number of reasons:

- EMF can work off models described in XML, which allows interoperability with other modeling tools. For instance, the digital representation of RIF could be used as the starting point for ProR, which sped things up considerably.
- EMF is modular. Halfway through the project, we switched to the EMF-based data model implementation from the ITEA-VERDE-Project (see Section 2.3). Thanks to the modular structure of EMF, this was straight forward.
- EMF takes care of many mundane tasks in GUI development.
- Rodin provides an EMF bridge. Even though Rodin is not based on EMF, there is a plugin that provides an EMF-based bridge to the Rodin data model. This plugin is actively maintained and makes it easy to integrate the data models from ProR and Rodin.

2.3 Requirements Interchange Format (RIF/ReqIF)

RIF/ReqIF 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.

RIF was created in 2004 by the “Herstellerinitiative Software” [HIS], a body of the German automotive industry that oversees vendor-independent collaboration. Within a few years, it evolved from version 1.0 to the current version 1.2. The format gained traction in the industry, and a number of industry tools support it.

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.

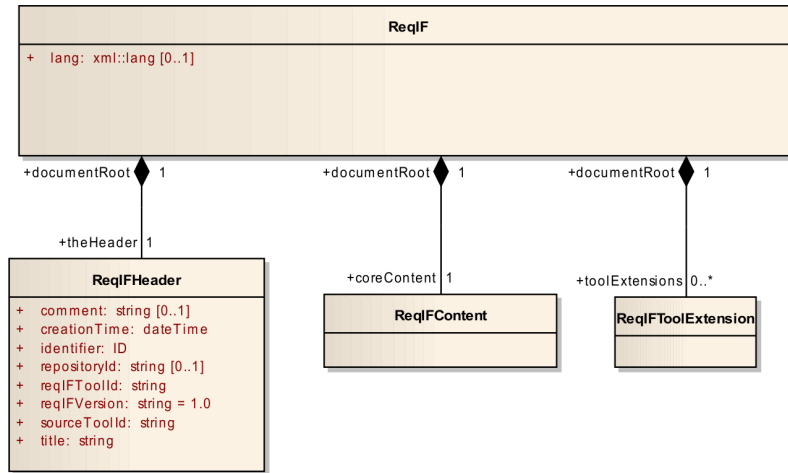


Figure 2: Top-Level Element of ReqIF 1.0

We were torn on whether to build ProR on RIF 1.2 or ReqIF 1.0. ReqIF is much cleaner, but is not yet finalized and there is no interoperability with industry tools. When we had the opportunity to incorporate the RIF 1.2-based ITEA-VERDE data model [Ver], we decided to use RIF 1.2.

To give an impression on the structure and documentation of RIF/ReqIF, Figure 2 shows the UML diagram of the top level ReqIF element, taken from the ReqIF specification. You can see that the top level element contains exactly one header, one content element and any number of tool extensions. Of course, this is just a small detail of the standard. The specification is much more elaborate.

3 ProR GUI

The GUI of ProR is shown in Figure 1. For Eclipse users, it should look immediately familiar. The left pane shows projects that can be expanded to show their content. The screenshot shows four RIF documents (“Requirements Documents” in the screenshot), together with some other elements of the project.

In the following, we use the ReqIF terminology. In ReqIF 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 Specifications.

The pane in the center contains a view of a Specification. It shows the hierarchical structure of the SpecHierarchies and their associated SpecObjects (“Requirement” in the screenshot). RIF supports multiple Specifications (“Specifications” in the screenshot). The columns of the main view can be customized (“View-specific Attributes”). The presentation of a SpecObject’s attributes can be customized with plugins that we call Presentations (“Custom Presentation with color highlighting”). Presentations are a ProR concept and are not part of RIF.

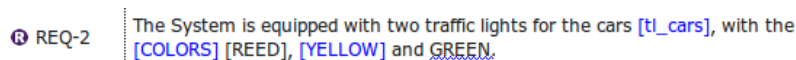
Below the main pane is a property view that shows the AttributeValues of the selected element (“Requirements Properties”). Note that the main view may not show all of the attributes of a Requirement, but the property view shows them all.

The right pane shows the structure of the RIF-document, grouped by Specifications, SpecObjects and SpecRelations.

4 Case Study

We built a specification of a traffic light system, consisting of 29 requirements, part of which can be seen in Figure 1.

The focus of the case study was to explore Rodin integration. Specifically, we wanted to evaluate the usability of the tool, and we wanted to find out how easy it was to create plugins that customize the platform.



```
REQ-2 The System is equipped with two traffic lights for the cars [tl_cars], with the [COLORS] [REED], [YELLOW] and GREEN.
```

Figure 3: A Requirement with Syntax Highlighting from the Event-B Model

Figure 3 shows the result of the syntax highlighting plugin. The highlighting is based on the research described in Section 1.2. Phenomena are extracted from the formal Event-B model and are highlighted in blue if they are marked by square brackets in the requirements text. Unmarked phenomena names are recognized and marked for inspection (e.g. “GREEN” in Figure 3). Text in square brackets that does not correspond to a phenomenon is not highlighted (e.g. “REED”).

This small case study was a success in demonstrating the potential of the platform.

4.1 Performance

As we are in the middle of development, we are not doing any performance tuning at this time. However, EMF is known for good performance “out of the box”. However, we are aware that it is not unusual to have RIF documents with thousands, even tens of thousands of requirements.

We built a test model with 1000 SpecHierarchy objects. The impact of the size was barely noticeable. With 5000 SpecHierarchies however, performance degraded significantly.

We believe that some performance tuning will be necessary eventually, but are content with the out-of-the-box performance.

5 Collaboration

We are actively reaching out to other public research projects and to industry. As mentioned, this resulted in the collaboration with Itea Verde [Ver], which sped up development considerably. We also talked to representatives of the ISYPROM project [ISY]. We also are in contact with members of the ReqIF standardization effort at OMG. We are actively compiling a list of industry contacts that we will use for validating the platform, once it reaches a more complete state.

6 Current State and Outlook

At this point, ProR only implements a small subset of the RIF data model. For persistence, we use the RIF core developed by the ITEA-Project Verde [Ver], whose contributors generously offered us to use their code.

Clearly, ProR is not ready for production yet. As of October 2010, ProR supports:

- SpecObjects, with plain text attributes only
- SpecHierarchies
- SpecHierarchyRoots (called Specifications in ReqIF)
- SpecRelations

This leaves out a number of advanced attribute types, a number of groups and access policies.

The immediate purpose of ProR is to support Deploy, which continues until February 2012. At the same time, we have a strong interest for ProR to survive the Deploy project and to gain momentum in industry. We are aware that Deploy and industry needs may differ at times. Nevertheless, we hope to be able to support all interested parties.

A beta version of ProR can be downloaded from our website. We hope to have a sufficiently complete version by the end of the year. In the meantime, we hope to recruit more collaborators and contributors. Please visit <http://www.pror.org> for more information.

References

- [Abr10] Jean-Raymond Abrial, *Modeling in Event-B: system and software engineering*, Cambridge University Press, 2010.
- [CJO⁺05] J. Coleman, C. Jones, I. Oliver, A. Romanovsky, and E. Troubitsyna, *RODIN (rigorous open development environment for complex systems)*, EDCC-5, Budapest, Supplementary Volume (2005), 23–26.
- [DDML97] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde, *GRAIL/KAOS: an environment for goal-driven requirements engineering*, Proceedings of the 19th international conference on Software engineering (Boston, Massachusetts, United States), ACM, 1997, pp. 612–613.
- [Dep] Project Deploy, *Industrial deployment of system engineering methods providing high dependability and productivity*, <http://www.deploy-project.eu/>.
- [Ecl] Eclipse, *The Eclipse Platform*, <http://eclipse.org/>.
- [EMF] EMF, *The Eclipse Modeling Framework*, <http://www.eclipse.org/emf/>.
- [FBWJ02] S. Faulk, J. Brackett, P. Ward, and J. Kirby Jr, *The CoRE method for real-time requirements*, Software, IEEE **9** (2002), no. 5, 22–33.
- [GJGZ00] Carl A. Gunter, Michael Jackson, Elsa L. Gunter, and Pamela Zave, *A reference model for requirements and specifications*, IEEE Software **17** (2000), 37–43.
- [HIS] HIS, *Hersteller initiative software*, <http://www.automotive-his.de/>.
- [ISY] ISYPROM, *Integrated systems and process modelling*, <http://www.isyprom.de/>.
- [Jac01] M Jackson, *Problem frames: analysing and structuring software development problems*, Addison-Wesley/ACM Press, Harlow England ;;New York, 2001.

- [Jas09] Michael Jastram, *Requirements traceability*, Tech. report, University of Southampton, 2009.
- [JHLR10] Michael Jastram, Stefan Hallerstede, Michael Leuschel, and Aryldo G Russo Jr, *An approach of requirements tracing in formal refinement*, VSTTE, Springer, 2010.
- [OMG] OMG, *Requirements interchange format (ReqIF) 1.0 beta 1*, <http://www.omg.org/spec/ReqIF/>.
- [Ver] ITEA2 Project Verde, *Validation-driven design for component-based architectures*, <http://www.itea-verde.org/>.