Modeling Requirements with RED

Harald Störrle

Department for Applied Mathematics and Computer Science
Technical University of Denmark
hsto@dtu.dk

1 Motivation

Shortcomings in Requirements Engineering (RE) are known to be the prime source of software development project failures. Better education can help improve this situation, but a critical prerequisite is adequate (and affordable) tooling for modeling and working with requirements specifications. The author has been teaching a major Requirements Engineering class at the Technical University of Denmark (DTU) for six years, and the need of tool support to do practical case-study work is quite obvious. We had prior experience with commercial tools (DOORS, RequisitePro, and JIRA), and we have tried different approaches to tooling in the course, including the "pragmatic" solution (spreadsheets and text documents), an open source RE tool (OSRMT, with massive customization), and a UML tool (MagicDraw, with extensions). Neither of them proved to be helpful.[1]

Among their disadvantages are their high cost (licenses, customization, learning), vendor lock-in, a hard-wired development approach, eclectic coverage of requirements concepts, and inconsistent terminology. In a nutshell, existing tools are focused on practical software production and specific customers while the concerns of education are less important. So, eventually, we decided to build our own tool, the Requirements Editor (RED). RED is specifically created to address the needs of students and teachers, but it has matured beyond a mere academic prototype: based on personal experience with RE tools in large development projects, we believe it outperforms many existing commercial products.

- **Comprehensive**: RED comes with a comprehensive set of features and concepts addressing many different practical scenarios.
- **Mature**: Through extensive class room usage, we have evolved RED to provide much needed features, usability improvements, and overall stability.
- **Understandable**: Consistency and learnability have been great concerns all along, and there is extensive teaching material available.

Thus, with RED, tooling will cease to be an obstacle in RE teaching, and, possibly, it will further RE in practice, too.

[1] See e.g. `http://makingofsoftware.com/resources/list-of-rm-tools` for a continuously updated list.

2 Tool capabilities

Previous versions of RED already provided a rich set of requirements editing capabilities, including classic textual requirements, stakeholders, goals, glossaries, sketch-style UML model fragments, and so on. Also, major tools like reporting, search, logging/status information, inspection support, and user-defined views were provided. Finally, all the usual qualities and UI capabilities of an Eclipse RCP application were available, which made RED a stable and easy to use tool from the start. However, the new version 3.0 of RED significantly extends the previous version [8], offering many new modeling features. See Fig. 1 for an incomplete overview of the current version of the meta-model.

– **Use Case Modeling**: RED now provides fully fledged Use Case modeling capabilities, focusing on the tabular view, which Dobing and Parsons describe as the most common form of use cases [4]. It does provide aspects such as cost and complexity to cover the most prevalent administrative aspects and support important requirement management activities.
– **Effort Estimation and Prioritization**: RED allows to aggregate cost/benefit estimates over any selection of functions, e.g., subsystems. It also allows sorting and contrasting of feature sets, and has built-in support for structured semi-automatic effort estimation based on the well-known Use Case Point method [3, 5].
– **Personas and Storyboards**: User centric systems require special attention to usage scenarios which are best captured using prototype sketching and personas [2]. RED allows to enter personas, and provide them with rich interactive storyboards specified as scenarios. Like Use Case scenarios, storyboards may be enacted.
– **Complex Scenarios with Enactment**: Scenarios may be used to enrich use cases and many other specification elements (e.g., connectors, actors, or personas). Beyond customary linear scenarios, we also offer a broad range of complex interaction operators such as the ones known from UML interactions (e.g., alt, par, loop). Steps may be specified textually, or by pictures so that a sequence of steps can be reported much like a graphic novel. Beyond mere specification, we also offer interactive enactment of scenarios, which is a valuable method of validating behavior [7].
– **Organization and System Structure Modeling**: a simple form of architecture modeling is provided with RED, that satisfies both the requirements suggested by ISO 42010 [6] (and its predecessor, IEEE P1471), and modeling of organizational structures.
– **Multi-File Projects**: with increasing project and team-size it becomes progressively more important to support distributed collaboration. A first step towards this is the introduction of multi-file projects which allows us to drastically reduce the number of edit conflicts. Tools for importing old file formats and interactively restructuring a project are included.

RED strives to be open in every sense, thus we allow to include any kind of documents into a RED project, acknowledging the fact that the most widely used requirements tools (in particular in the early phases), are Word, Excel, and PowerPoint. This kind of document can be stored in a RED project, and opened from within RED (as far as the respective tools are available).

3 Project History and Usage Experiences

Faced with these requirements, we decided to create RED using the Eclipse Rich Client Platform (ERCP). Since 2011, seven students have contributed to RED through their MSc-thesis projects, and three more are currently on-going, adding up to almost 300 ECTS points worth. A student programmer is employed to coordinate and integrate the contributions, fix bugs, and add small features not fitting into the larger projects.

A first version was deployed in September 2012 and since then, we have used RED continuously in our RE courses.[2] By now, RED has been used by students for several thousand hours. The feedback received there was used to improve the tool, and while initial reactions were mixed, they have gradually improved. Students now use RED without problems.

The latest version has not been field tested yet, but developer tests suggests no decrease in stability or usability, while the feature set has expanded considerably. We believe that RED is now relatively close to commercial solutions with regards to capabilities and qualities.

4 Architecture and Implementation

RED has been created using the latest Eclipse platform available at the time (Eclipse 3.7 "Indigo"). RED is organized in a set of components ("features" in Eclipse terminology, see Fig. 2), each providing specific bundles of concepts and capabilities: the Core module provides the main UI of the application and the back-bone of the meta-model. It supports a number of feature-modules, such as Glossary, SpecificationElements and Help. We have also reused a number of third-party plug-ins, including EPF RichText and AgileGrid, that increased the code reuse ratio.

The main rationale behind for using Eclipse is its proven ability to create rich, cross-platform applications. Due to its plug-in-architecture, significant leverage through reuse was achieved. Adopting a popular framework, we also ensured maintainability and long-term development.

One particular highlight in the release currently under development are the visual editors. They will present an easy to use front-end for specifying many of those elements of a requirements specification that we have come to expect as being primarily visual, e.g., use case and goal models. They will supplement the existing tabular and form-based editors. Adding a visual front end will allow both faster modeling and more effective communication of specifications.

[2] In this course, there are 50-60 students per year, working in groups of 4-6 for 13 weeks, spending approximately 25 hours per week for the course.

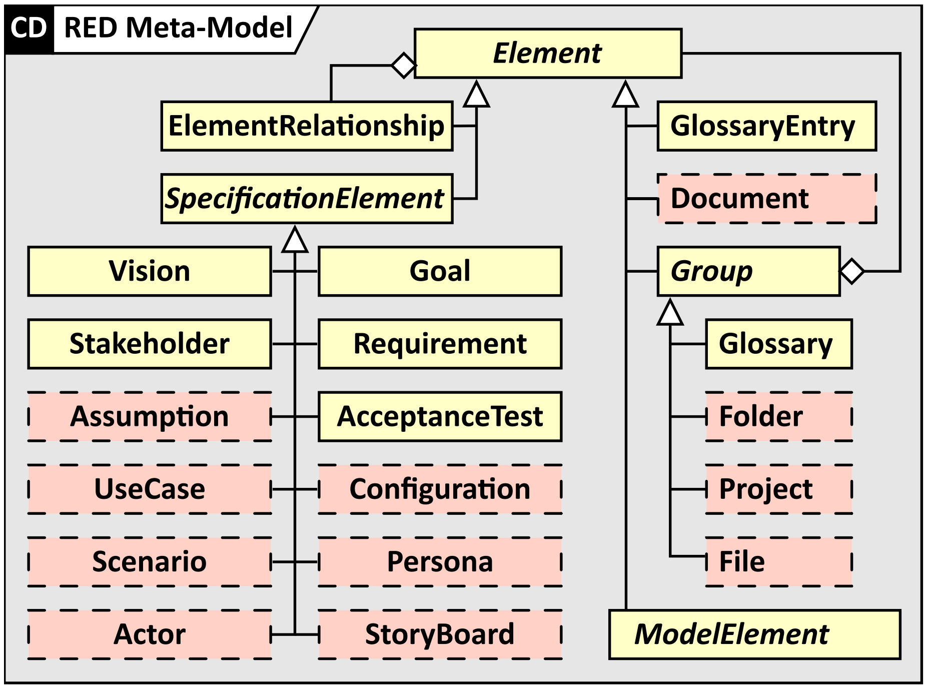

Fig. 1. Excerpt of the meta-model of RED: meta classes highlighted in red with dashed outlines are new or substantially extended in this version of RED.

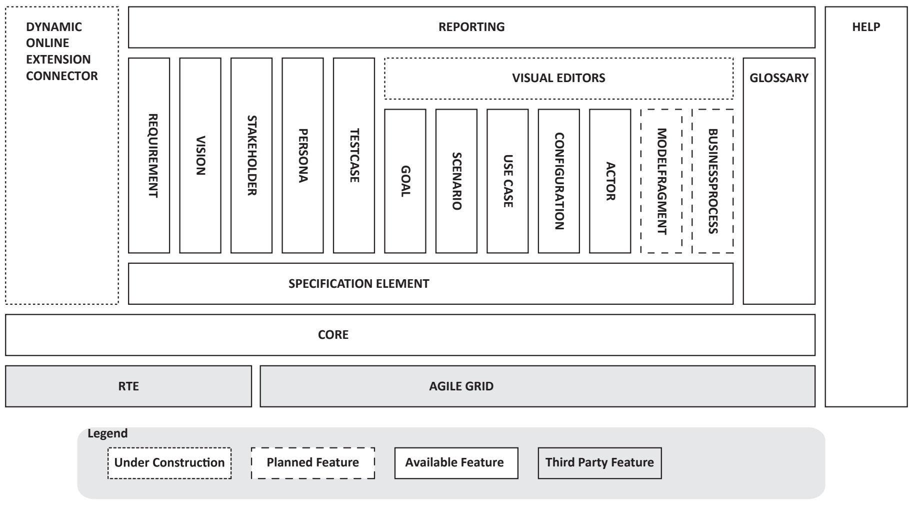

Fig. 2. High-level view on the architecture of RED: boxes represent Eclipse features (i.e., OSGI bundles).

Another highlight is the Dynamic Online Extension Connector which is a novel feature allowing to seamlessly integrate features at run-time via web-services. This is useful to provide features which are difficult to integrate for technical reasons (e.g., they are based on a different technology), that require more resources than are readily available on a standard client desktop, that must be deployed late, or that must be maintained and updated independently. It is also a way to allow a wide spectrum of extensions to be provided by third parties, avoiding IP issues. One example of such a service is Hypersonic [1].

5 Ongoing and Future Work

We currently focus on providing a few missing features such as visual editors, connectors for other formats such as ReqIF, and a teamwork server to support distributed work, easy to use version control, and controlled natural language checking facilities. Also, the recent set of additions prompts us to consolidate the meta model and software architecture of RED once more. Further features are currently postponed due to resource shortage, including support for star-card based approaches as used for "agile" requirements.

We are working on legal issues which, when resolved, will allow us to release RED under a liberal licensing scheme. Clearly, this is the cornerstone both to attracting outside contributions to further evolve RED, and to allow its usage in industrial projects where it will realize its true potential.

References

1. Vlad Acretoaie and Harald Störrle. Hypersonic: Model Analysis and Checking in the Cloud. In Dimitris Kolovos, Davide DiRuscio, Nicholas Matragkas, Juan De Lara, Istvan Rath, and Massimo Tisi, editors, *Proc. Ws. BIG MDE*, 2014.
2. Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.
3. Mike Cohn. *Agile Estimation and Planning*. Prentice Hall PTR, 2005.
4. Brian Dobing and Jeffrey Parsons. How UML is used. *Comm. ACM*, 49(5):109–113, 2006.
5. Stephan Frohnhoff and Gregor Engels. Revised use case point method-effort estimation in development projects for business applications. *Proc. CONQUEST*, 2008.
6. ISO/IEC/IEEE 42010:2011 - Systems and software engineering - Architecture description. Technical report, ISO, 2011. Retrieved 2014-07-06.
7. Keith Phalp and Karl Cox. Using Enactable Models to Enhance Use Case Descriptions. In *Proc. Intl. Ws. Software Process Simulation Modelling (ProSim, co-located ICSE)*, 2003.
8. Harald Störrle and Maciej Kucharek. The Requirements Editor RED. In Bernard Carré, Houary Sahroui, and Harald Störrle, editors, *ECOOP, ECSA and ECMFA 2013: Joint Proceedings of Tools, Demos & Posters*, pages 32–34, 2013. DTU Technical Report 2014-01.