RED - A Requirements Engineering Tool

We're going open source SOON!

Harald Störrle (hsto@imm.dtu.dk)

DTU Compute
Department of Applied Mathematics and Computer Science

Objective

Each year, about 60 students take the "Requirements Engineering" course taught by Harald Störrle at the Technical University of Denmark (DTU). The course covers a wide variety of techniques in a hands-on fashion, so that tool support is dearly needed.

The main goals of such a tool are (1) to help teaching RE and (2) to support defining and exploring innovative concepts not usually found in commercial tools.

Existing Tools

Every available tool covers only a small subset of the many techniques taught in this course. They all use different (and often inconsistent) terminology, were hard to customize and extend, and/or are too expensive to purchase.

After several failed attempts to use preexisting commercial and open source tools, we decided to build our own.

Required Quality Attributes

With a view to the intended audience, usability, stability, and portability are clearly important aspects. Performance, on the other hand, is of lesser importance, and safety/security are clearly not relevant.

Also, given the limited scope of any one contribution (e.g., a MSc-thesis), it is clear that this is a long-term project, so that maintainability would be a major concern.

Required Features

In order to suport the pedagogical goal, the tool must procide a large list of features:

- editors for the techniques syllabus,
- consistent terminology,
- support collaboration of small teams,
- support for Fagan-style inspections.

Of course, students also expect all the "usual" features, such as undo/redo, auto save, visual UI, reporting, online help, etc.

Architecture & Technology

Since it was clear form the outset that fulfilling the RED vision would be a long-term project, we had to make a number of technological choices geared towards this context.

Turning RED into an open source project eventually will also raise a number of legal questions concerning intellectual property which we are currently investigating.

Language

We chose Java as our language environment, mainly due to the predominance of Java in academic environments, this is effectively the only choice for a software ecosystems for this kind of project where we rely on a contionuous supply of students to carry on the work.

Also, platform independence is a must for a class-room tool, where all major platforms are in use.

Platform

In order to address the requirements stated above, we decided to use Eclipse RCP as our platform for implementing RED: it offers high levels of stability and leverage through reuse and automation/code generation.

At the same time, substantial learning effort is required, but Eclipse is a widely used platform, and utilizing the recent Eclipse platform 4.3 (Kepler) has lowered this entry barrier.

Version Control / Issue Tracking

We now use GIT hosted at Atlassian BitBucket to store RED, featuring an issue tracking system based on Jira. We use STAN4J for tracking code quality metrics.

Build/Make/Deploy Process

RED comes with build-setup for development using Eclipse as an IDE, but also with a Tycho-build script, which is an extension using Maven.

The RED tool (v2.0)

Navigating a specification

Specification projects may be browsed with a tree navigator handling multiple project files, or a topological browser. The tree view offers sorting, filtering, and a full-text search, where results are shown in a sorted tree-view. References to other elements are clickable, and there are back/forward buttons to quickly move between recently viewed elements.

Requirement Secification Elements

Probably the most important feature of RED is its wide range of supported requirement specification styles, including:

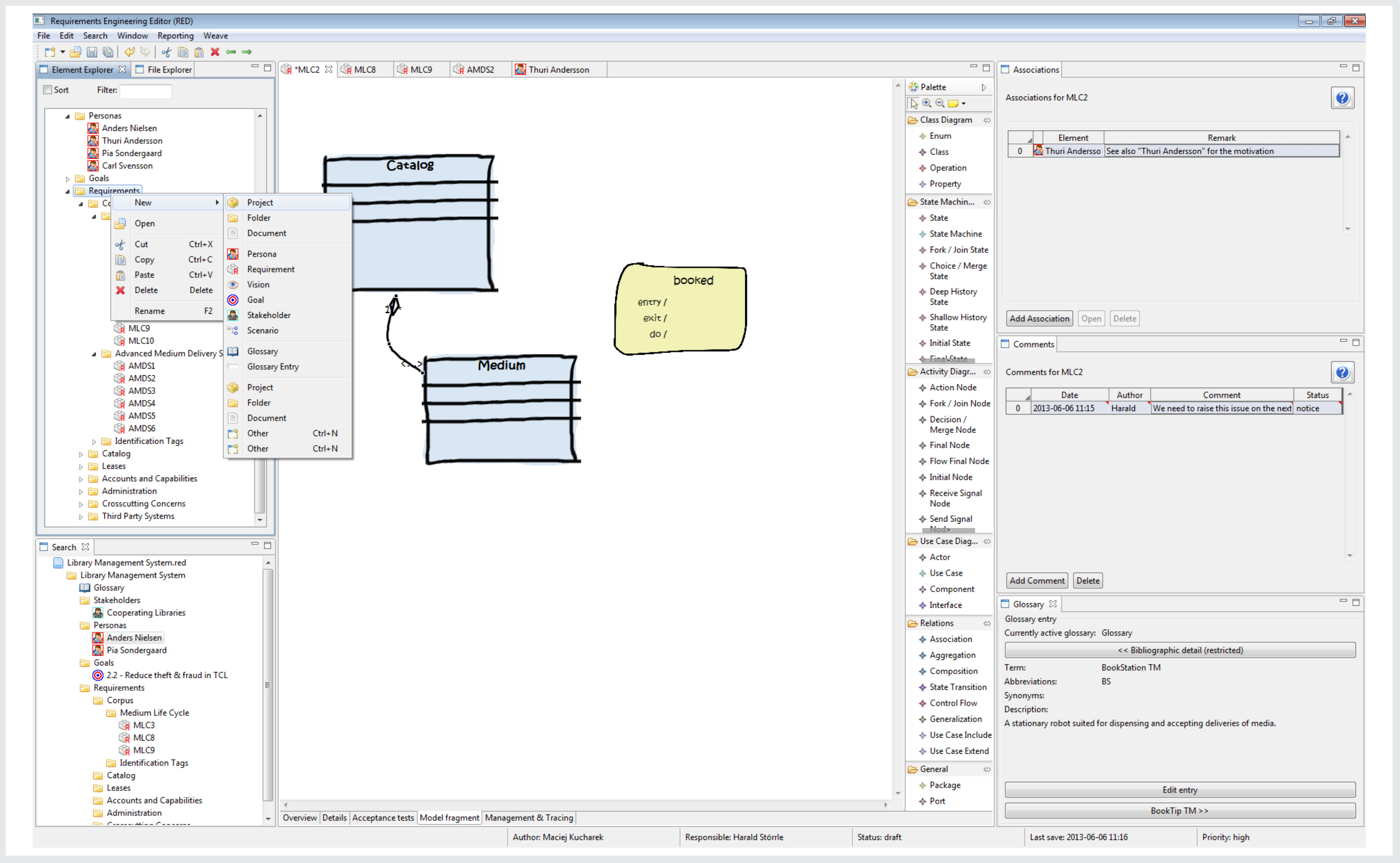
- **Project** as a container for a specification;
- **Vision** of the specification as prose;
- **Goals** in textual/structured form; a graphical goal model editor is currently missing.
- **Stakeholders** are described by prose and a number of properties such as urgency, power, kind etc.
- **Personas** are described in strctured text.
- **Scenarios** may be described in prose, in photo novel/comic style, or as structured text. Structured text may be enacted, which allows a behavior preview.
- **Requirements** may be described using prose, test cases, and model fragments.
- **Glossaries** provide consolidated project terminology. When terminology items are used in any other piece of text, they can be made to refer to the glossary items.
- **Documents** are arbitrary RTF texts that may be included in a requirements specification, including e.g. appendices.

Reporting

One of the most important functions to students (and not just them), is the ability to create reports from requirements projects. Currently, only HTML-export is supported, but this is easily loaded by word processors such as MS Word, and then stored in any format supported there, e.g., DOCX or RTF.

Commenting/Reviewing

All elements afford commenting. A comment records a timestamp and the author. Arbitrary parts of a specification may be locked with a password against changes. A partial lock may be issued, allowing only the addition of comments, as whe passing on the specifications to an inspection. Comments may be exported separately.



Management

All elements of a requirements specification are equipped with uniform management and tracing information, such as version number, timestamp, history, change comment, references to previous, superseeded elements, author/changer and other responsibilities, priority, etc.

Tracing

All elements of a requirements specification are equipped with uniform management and tracing information, such as version number, timestamp, history, change comment, references to previous, superseeded elements, author/changer and other responsibilities, priority, etc. All elements may be linked; links are navigable in both directions.

Requirements & Model Fragments

Traditional textual requirements are captured in a structured way, using rich text editors for the various aspects. Requirements also allow to specify structured test cases, references to external documents, and even fragments of UML models (class, activity, use case, deployment, and sequence models). The Fragment editor provides a sketchy view to invite changes and discussions.

On-line Help

The on-line help system integrates material from the course's lecture notes, providing help primarily on the specification techniqes rather than on tool usage.

Status Bar

The status bar shows the save-status, roles (author, responsible, changer), lock status, QA and completion status, and specification item priority.

Model Fragment Weaving

Sets of model fragments may be combined into more complete models by a semi-automatic weaving process. Export to XMI is not provided yet.

Ongoing Work

RED 1.0 has been used in a RE course in 2012, resulting in a large number of small issues. In the 2013 course, reception of RED 2.0 was much improved, but new challenges arose, too.

First, we want to make RED an open source project to take advantage of community contributions. We have established procedures and staff to deal with Issue Tracking, Building, and Deployment, so we're almost there!

Second, many novel features are still missing (e.g., controlled natural language checking, templated text, visual goal/context modeling). The recent reengineering exercise tremendously helps us addressing these.

Third, RED does not currently support cooperative and distributed work. Two ongoing complementary projects provide a teamwork server and client to allow collaborative requirements engineering.