A Modular Requirements Engineering Framework for Web-based Toolchain Integration

Robert Darimont, Wei Zhao
Respect-IT S.A.
Louvain-la-Neuve, Belgium
email: {robert.darimont,wei.zhao}@respect-it.be

Christophe Ponsard, Arnaud Michot
CETIC Research Centre
Charleroi, Belgium
email: {christophe.ponsard,arnaud.michot}@cetic.be

***Abstract*—Requirements Engineering (RE) tools and more generally the whole Software Engineering toolchain follow the strong trend towards web-based interface. This allows the analyst to use their tools in a "Software as a Service" mode either from a local company server or directly in the Cloud. Such deployments also ease toolchain integration by connecting their respective API through secured web-services, possibly using specific software lifecycle interoperability standards. In this tool demonstration, we illustrate the results of the rewrite process of a major Requirements Engineering tool towards this purpose. Our tooling has the following key features: (i) it supports rich requirements models based on goal-oriented RE, (ii) it is implemented as a collaborative concept server based on Eclipse Modelling technology and (iii) it exposes a REST interface supporting model building, diagram edition, history retrieval, snapshot management, collaborative mode, user authentication and project management. The following scenarios will be demonstrated (1) collaborative edition of a shared RE model, (2) rich service composition with application lifecycle management tools and (3) easy web-component integration in third-party web interfaces.**

I. Background and Motivation

The landscape of requirements tools is mostly composed of two kinds of complementary tools:

- *Requirements Engineering* (RE) tools support the whole process to produce high quality requirements, this includes elicitation from various sources, analysis (e.g. building goal trees or performing obstacle analysis [1]), specification (e.g. using templates like Volere [2]) and validation (e.g. review of goal models or derived documents). There are either specialised RE tools relying on a strong model like Objectiver [3] and RE-TOOLS [4]. Many recent system engineering and enterprise architecture tools also propose some form of RE support.
- *Requirements Management* (RM) tools focus on providing a reliable and adequate persistent storage for import/export, traceability, versioning and change management. Examples of tools are DOORS [5] and Reqtify [6].

.

The classical design of RE tools is a 2-tier desktop application with a repository either locally or as shared database for working on multiple projects and in team. Such repositories can be synchronised with a RM tool which is usually a 3-tier system composed of an interface (desktop or web), an API for managing requirements and an efficient database.

The strong trend is to move to SaaS (Software as a Service) paradigm. Reducing client requirements to a recent web browser strongly eases installation and updates. It also removes infrastructure needs in case the companies agree to use an external Cloud. However this later scenario requires to address non-trivial security and confidentiality requirements.

Beyond those operation benefits, a SaaS deployment also provides more efficient support of RE activities by easing integration and teamwork. In addition, it enables quite powerful integration scenarios such as the inclusion of "live" RE artefacts (like references, definitions, diagrams, model queries,...) in other Application Lifecycle Management (ALM) tools at design, testing or maintenance stages.

In this tool demo, we illustrate such scenarios based on the SaaS migration of Objectiver tool which is dedicated to KAOS (Keep All Objective Satisfied), a Goal-Oriented Requirements Engineering method [1].

II. Architecture of a Web-based RE Tooling

Two major requirements for moving to a SaaS model are:

- *on the client side:* a powerful web-based framework for diagram edition and model navigation
- *on the server side:* a reliable application server offering a rich API for model-edition and a concept repository.

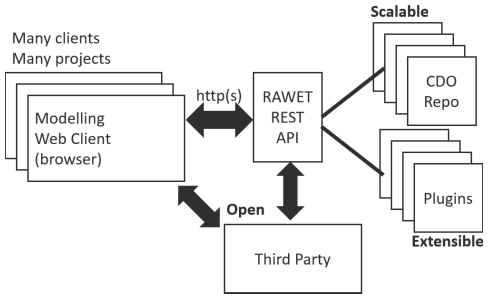

Fig. 1. Architecture of the Web-based RE Tooling

Our design relies on two key enablers. On the client side, there are now mature JavaScript diagram libraries [7]. On the server side, the Eclipse Modelling Framework [8] is an industry-strength meta-modelling framework widely used in the Eclipse ecosystem. Concept repositories like Connected Data Objects (CDO) and EMF-Store have been developed on

2332-6441/16 $31.00 © 2016 IEEE
DOI 10.1109/RE.2016.49

IEEE computer society

RE 2016, Beijing, China
Posters and Tool Demo Paper

top of it. We selected the former technology which supports fine-grained locking and versioning while the later relies on a check-in/out mechanism. We bundled it as a console application with a generic REST API as middle-tier. The resulting architecture is depicted in Figure 1 and is composed of:

- *various clients*, possibly embedded in third party tools.
- *a REST API* providing services for model and diagram edition, history, snapshots, user authentication and project management [9]. It also enables third party integration of RE artefacts extracted from the repository, not only as data but also as embedded web-content.
- *a back-end* composed of the model repository and a collection of plugins enabling both web-services and user interface extensions.

III. Some Interesting Scenarios

A. Collaborative Modelling Scenarios

The concept repository supports locking at different levels (concept, diagram, package) but also versioning and history browsing, enabling the development of different collaborative models. By default, the tool provides a permissive collaboration model where concurrent editions will just overwrite each other and trigger warnings (such as "the concept you are editing has been deleted"). The availability of the concept history allows one to easily identify the concurrent editing user and engage in a discussion with him. Beyond this, other collaborative models are also possible, e.g. automatic lock when editing a diagram resulting in read-only mode for other users accessing it.

B. Service Level Integration Scenarios

Two kinds of scenarios are supported: first, our REST API can be used to perform "CRUD" operations on the model and its representations. This means concepts like requirements, expectations, or obstacles can be queried, imported, added in diagrams or even laid out using dedicated algorithms. This scenario can be used to synchronise requirements with an external tool. Second, it is also possible to invoke external services by wrapping them into a plugin. Base on this, external tools can be notified of changes in real-time.

From a security perspective, those facilities might cause a threat. For this reason authentication is required through a specific service that returns a security token that can be used to implement single sign-on within the target tool chain.

C. User Interface Level Integration Scenarios

Going one step beyond interaction between web-services, web-components can be integrated in two ways. First, specific editors can be added as plugins in order to edit specific attribute types. For example, Figure 2 shows an editor for specification under the Gherkin format. Second, our web-components can be embedded into external user interfaces. A common example is to embed a read-only goal diagram on a wiki page as shown in Figure 3. The goal diagram is guaranteed to be up to date and double clicking on it will lead to the full editor (if the user has the access rights).

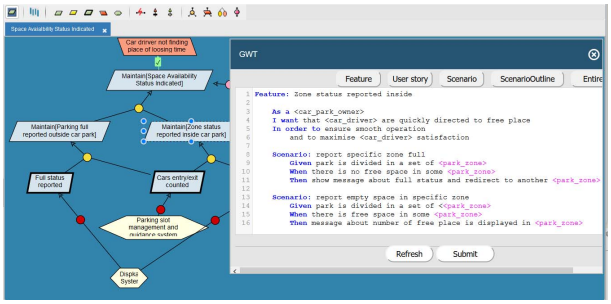

Fig. 2. Integration of an external specification editor

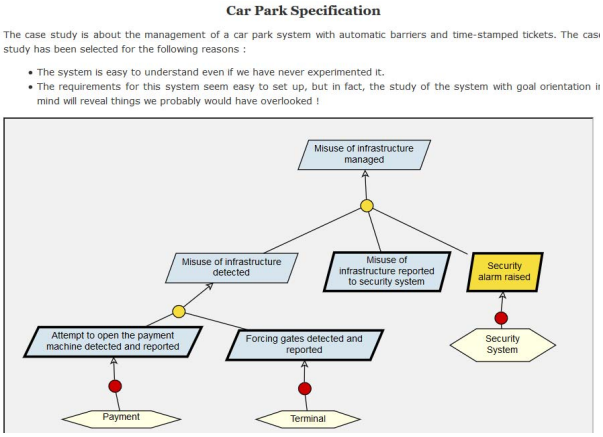

Fig. 3. Integration of an external specification editor

IV. Status and Next Steps

We are currently focusing on tool integration inside companies. A version hosted on a Cloud infrastructure is under development. An integration SDK is also being documented. Our future work will extend our support towards standard RE exchange formats and enrich the platform extensibility.

Acknowledgement

This work was supported by the Walloon Region through a software feasibility study. We warmly thanks Huawei tooling R&D dpt for challenging us with tricky integration scenarios.

References

[1] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, March 2009.

[2] S. Robertson and J. Robertson, *Mastering the Requirements Process (2Nd Edition)*. Addison-Wesley Professional, 2006.

[3] Respect-IT, "The Objectiver Goal-Oriented Requirements Engineering Tool," http://www.objectiver.com, 2005.

[4] S. Supakkul and L. Chung, "The RE-Tools: A multi-notational requirements modeling toolkit," in *20th IEEE Int. Req. Eng. Conf.*, Sept 2012.

[5] IBM, "DOORS," http://www-03.ibm.com/software/products/en/ratidoor.

[6] Dassault Systems, "Reqtify," http://reqtify.com.

[7] H. Ed-douibi, "10 JavaScript libraries to draw your own diagrams," http://modeling-languages.com/javascript-drawing, April 2015.

[8] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.

[9] C. Ponsard, A. Michot, R. Darimont, and W. Zhao, "A generic rest api on top of eclipse cdo for web-based modelling," EclipseCon France, Toulouse, June 2016.