Sysiphus: Combining system modeling with collaboration and rationale

Timo Wolf, Allen H. Dutoit
Institut für Informatik, Technische Universität München
Lehrstuhl für Angewandte Softwaretechnik
http://sysiphus.informatik.tu-muenchen.de/
{wolft,dutoit}@in.tum.de

1. Introduction

Sysiphus, a suite of tools developed at the Technische Universität München [3,1], enables users to develop and share system models and to discuss, justify, and collaborate within the same environment. On the one hand, users create model elements (e.g., use cases, subsystem decompositions, class models, test specifications) and organize them into documents. On the other hand, they discuss models informally using discussion threads, extract design decisions into more formal rationales, and plan and track their work using action items. Discussion threads, rationales, and action items are linked to the relevant model elements, capturing the context in which decisions are made. The originality of sysiphus is that it puts equal importance on system modeling, collaboration, and rationale.

2. Rationale model overview

Rationale, the justification behind decisions, is usually not captured and only exchanged informally. While requirements and design decisions are captured in system models, their rationale is gradually lost as participants move to other projects or as their memories fade. The assumption behind rationale management is that externalizing rationale, making it available to project participants, and structuring it for long-term use can increase the quality of decisions, increase trust and consensus among stakeholders, and facilitate changes to existing systems [5]. While the usefulness of rationale management has been shown in specific cases, overhead associated with training participants and formalizing knowledge remain significant obstacles [1].

To represent rationale we use an *issue model* as proposed by argumentation-based rationale approaches [1]. Issue models represent the individual decision making elements that lead to a decision as individual nodes and their relationships with edges. Many different models have been proposed, including IBIS (Issue Based Information System, [2]) and QOC (Questions, Options, Criteria, [4]), to name the principal ones. We use a refinement of QOC that includes the following elements:

- *Issues* represent needs to be solved for the development process to proceed. Issues can indicate a design issue, a request for clarification, or a possible defect.
- *Options* are solutions that could address the issue under consideration. These include options that were explored but discarded because they did not satisfy one or more criteria.
- *Criteria* are desirable qualities that the selected option should satisfy. In our model, criteria are nonfunctional requirements, system design goals, or test criteria.
- *Assessments* represent the evaluation of a single option against a criterion. An assessment indicates whether an option satisfies, helps, hurts, or violates a criterion.
- *Arguments* represent the opinions of individual stakeholders, in particular, about the relevance of an issue or the accuracy of an assessment. By arguing about relative merits of options, stakeholders build consensus and converge towards a solution.
- A *decision* is the resolution of an issue representing the selected option. Decisions are already implicitly captured in the system models during development. We only need to capture the relationship between decisions and their corresponding rationale.

Issues can be linked to any number of model elements. Similarly, a single model element can be linked to any number of issues. Discussion threads and action items can be linked to both model and rationale elements. Model elements, one the one side, and discussion threads, rationale, and action items, on the other side, are viewed next to each other, enabling the user to go back and forth between the system model and its justification. Unlike Design Space Analysis [4], the justification activity does not drive the modeling activity. Conversely, rationale is not buried as an afterthought in the form of annotations contained in the model.

3. Sysiphus

The design goals of sysiphus are to provide an integrated solution to manipulate system models and rationale, embedding only minimal process specific knowledge. This enables us to adopt different development processes for different projects and to use rationale for a broad range of activities. Sysiphus is composed of a suite of tools centered on a repository, which stores all models, rationale, and user information. The repository controls access and concurrency, enabling multiple users to work at the same time on the same models. Model elements are organized into documents that provide activity specific views into the repository. For example, a requirements specification document can be defined in terms of sections that include use cases and nonfunctional requirements. Moreover, the same model element can also appear in different documents. To support any project needs, the documents and document structures can be created, modified and deleted.

Users access models either through a web interface or a graphical user interface. Both interfaces access the repository through a common model layer that defines the type of elements and links. The model layer can easily be extended to accommodate new element and link types, as dictated by the needs of a project. The model layer in tern uses the sysiphus repository for persistency and for pushing changes to other clients.

With the web interface users can access the repository from a variety of environments (e.g., lab, home, office) without the installation of additional software. It is divided into two parts, one for accessing and modifying the system models, the other for accessing the discussion threads, rationale and action items. Users can trace from the system models to relevant collaboration models and back.

The graphical user interface provides a UML view over the models, adopting the look and feel of typical UML modeling tools. The graphical user interface is typically used for drawing diagrams and, with drag and drop interaction styles, offers more intuitive ways to specify relationships among model elements than the web interface.

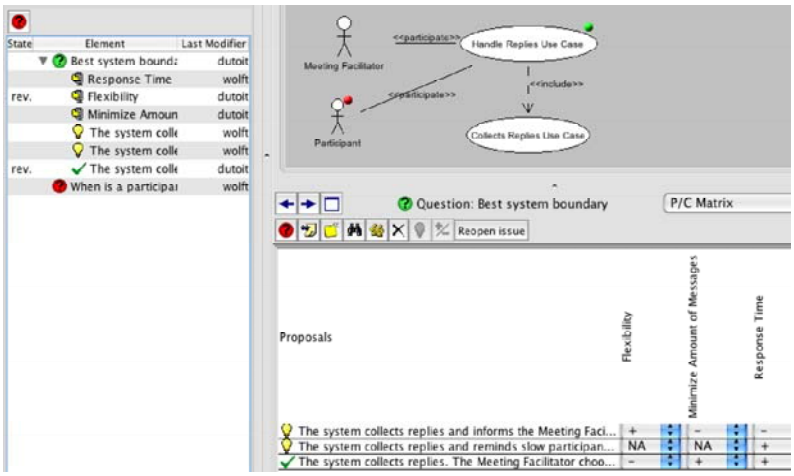

Figure 1: *Graphical user interface overview: Diagrams (top right), issues (left) and a content pane (bottom right) provide alternate ways to access and create links between system elements and rationale.*

References

1. S. Buckingham Shum & N. Hammond, "Argumentation-based design rationale: what use at what cost?" International Journal of Human-Computer Studies, vol. 40, pp. 603–652, 1994.
2. J. Conklin & K. C. Burgess-Yakemovic, "A process-oriented approach to design rationale," Human-Computer Interaction, vol. 6, pp. 357–391, 1991.
3. A.H. Dutoit, B. Paech. "Rationale-based Use Case Specification" Requirements Engineering Journal, vol. 7, pp. 3–19, Springer Verlag, Mar 2002.
4. A. MacLean, R. M. Young, V. Bellotti, & T. Moran, "Questions, options, and criteria: Elements of design space analysis," Human-Computer Interaction, vol. 6, pp. 201–250, 1991.
5. T.P. Moran, J.M. Carroll (Eds.) Design Rationale: Concepts, Techniques, and Use. Lawrence Erlbaum, Mahwah, NJ, 1996.
6. T. Wolf, A.H. Dutoit "A rationale-based analysis tool." 13th International Conference on Intelligent & Adaptive Systems and Software Engineering, July 1-3, 2004, Nice, France