The Challenges of Tool Integration for Requirements Engineering

Lisa K. Meisenbacher

Siemens Corporate Research, Inc.

lisa.meisenbacher@siemens.com

Abstract

Requirements elicitation and management have been well documented using the Unified Modeling Language (UML) Business Object Models approach (Berenbach, May 2004, October 2004, 2003). There are many software tools that either fully or partially support UML standards. Some tools are more popular than others, and all have their own advantages and disadvantages. Integration of commercially available software products is something of a misnomer. This paper describes techniques that facilitate tool integration for requirements management and maintenance. Specific examples detail the capabilities (and/or limitations) of existing commercial products that a requirements engineer may use to model a business process and elicit detailed requirements. In addition, the case study will describe three challenges encountered with tool integration: traceability, hierarchical organization, and tool maintenance.

Keywords: requirements engineering tools, CMMI, tool integration, COTS, traceability, tool maintenance

1 Introduction

Requirements elicitation and management involves using software tools. Many of these tools are commercially available, some are not. The requirements engineer must have fully integrated tools available to provide consistent and uniform solutions for requirements engineering problems. The challenge of tool integration becomes difficult in many instances since many use proprietary API's and do not easily work together with each other.

Requirements engineering tools are becoming increasingly applied in globally distributed environments. Companies can be segmented across different subsidiaries, divisions and geographies. Cross-functional teams are being formed in order to facilitate collaboration in requirements elicitation and management. This trend highlights the need for disparate tools to be fully integrated.

Moreover, there is no single or uniform approach to requirements engineering tool use. In addition, as organizations become CMM and CMMI compliant with their requirements engineering processes, seamless tool integration becomes critical for establishing and maintaining traceability of requirements.

This paper will illustrate an approach that uses COTS products to provide a CMMI compliant and integrated process to facilitate requirements elicitation and management.

The case study described below focuses on describing solutions for three challenging areas: bidirectional traceability, hierarchical organization of requirements, and long-term tool maintenance.

2 COTS Vendor Landscape

There are many commercial off-the-shelf tools available for different stages of the requirements engineering process. Some of the more well-known ones include Caliber (Borland Software Corporation, 2005), DOORS (Telelogic AB, 2005), Rhapsody (I-Logix Inc., 2005) and Requisite Pro (Rational Corporation, 2005). The results of a Google search on requirements engineering tools show there are more than fifty COTS products currently available. For better or for worse, there are many tools to choose from. Although websites such as Volere and Easyweb detail the capabilities of each of the COTS tools, neither describe how or if any of these tools work together or have synergies available that facilitate development, traceability or requirements maintenance across different product suites (Requirements Engineering Tool Vendors and Freeware Suppliers, 2005; Volere, 2005).

Some vendor websites highlight specific areas of tool integration with little or no detail regarding the "degree" of integration or just the overall capabilities and limitations. My experience with one particular Siemens operating company indicates that requirements tool integration "gotchas" are not uncovered until the analyst or team of analysts is "knee-deep" into the product functionality and the project is already committed to a particular tool or suite of tools.

With that in mind, this paper will detail the findings of one particular on-going project that uncovers both the benefits and limitations of the following commercially available tools: Rational Rose, Doors, Rose Integration and Doors DXL. Additionally, the details of the case study will also provide the reader(s) with valuable insight into how much of the product integration is "out-of-the-box" and how much you must build on your own or buy from a third party vendor.

3 Challenges

3.1 Traceability

Bidirectional traceability is critical for CMMI compliance. Tool integration must provide the ability to trace Rose artifacts to and from their respective requirement(s). In this case study, the business object model was created in Rational Rose. The text requirements reside in an enterprise Doors database. Rose Integration provided the mechanism for bidirectional traceability. Rose Integration provides the ability to link each Rose artifact to the respective Doors requirement(s) as part of the core product functionality.

Programming with Doors DXL was necessary in order to automate the creation of these links. Programming proficiency with Doors DXL typically requires additional training in order to acquire the in-house expertise necessary for customization. DXL scripting requires minimal training for an experienced C++ developer but more extensive training for novice programmers or those new to C++.

3.2 Organization

The Rose Integration software is used to create a surrogate Doors module that contains the bidirectional links. The Rose business object model contains a hierarchical representation of the requirements. Parent-child relationships are used to accurately represent the relationships between the artifacts. The Doors surrogate module does not have any knowledge of the relationships between the Rose artifacts.

The Rose model is imported into the Doors "surrogate" module with a flat hierarchy. Specifically, all hierarchical information from the Rose model is lost when the model is imported into Doors using the "out-of-the-box" functionality of Rose Integration. All UML diagrams that detail the artifact relationships are also lost. Significant effort and programming are needed to rebuild the original hierarchy in order to represent the original Rose model in a meaningful and useful fashion. Section four will describe the solution used to create a hierarchical module.

3.3 Maintenance

Maintaining the imposed hierarchy of the Doors surrogate module poses some long-term tool support issues. Although Telelogic recommends customizing solutions with DXL programming, they have confirmed that the proposed solution outlined in this paper voids the product warranty and will not be supported. In essence, the product vendor will only support a solution that is unusable for the purpose of this case study.

Long-term requirements maintenance considered over the full lifecycle of a software product(s) requires tool flexibility as well as short-term and long-term support from the product vendor regardless of whether it is an off-the-shelf or a customized solution.

4 Challenges

The Rose model was imported into Doors using the Rose Integration software (v2.8). DXL scripts were written to recreate the original Rose model hierarchy in the surrogate module. The Rose documentation fields were also migrated into the surrogate module during the import step. Additional DXL scripts were written to parse the description fields of the Rose model and create bidirectional links from Rose to the surrogate module and then to the stakeholder requests located in a separate Doors module. A convention was adopted for the direction of the links (outgoing versus incoming) since the Telelogic definition of link direction was ambiguous. Furthermore, the hierarchical surrogate serves as the focal point for navigating between the text requirements in Doors and the Rose business object model.

A working prototype is in place that provides a meaningful framework for automatically generating a detailed systems requirement specification (SRS) from the requirements. The business object modeling of the requirements is on-going. As new objects are created, modified and/or deleted in Rose, the surrogate module is regularly updated using the Rose Integration tool.

The DXL script that creates the hierarchy needs to be invoked each time the Rose model is updated and these updates are sent to the surrogate module. This extra step is necessary because the Rose Integration tool is agnostic of the imposed hierarchy on the surrogate without the DXL script. Furthermore, the Rose Integration update process has only been used to send updates from Rose to Doors for this case study. Although Rose Integration has the ability to create new model elements in Doors and export them to back to Rose, this feature has not been used as part of the case study solution (Telelogic AB, 2005).

The reader should be aware of an alternate third party solution. Galactic Solutions Group has also developed a proprietary solution for imposing a hierarchical structure on a Doors module. The RM-Int product exports the entire Rose model hierarchy (including UML diagrams) into a Doors module. Although this product is fully

supported by the vendor, it is not yet commercially available (Galactic Solutions Group, 2005). This was also a contributing factor for developing a customized solution specific to the Siemens operating company.

5 Summary

A case study solution was presented that outlines the methodology used to create a customized solution for COTS integration specifically for requirements management. The prototype developed will be used to automatically generate systems requirements specification for a future Siemens software product.

There are currently many COTS products that provide a rudimentary foundation for integrating requirements engineering tools. However, many tools require special customization to meet the particular requirements and business needs of a project.

Product vendors should carefully consider the importance of tool integration as they continue to implement new releases and next generation products for requirements engineering. From a technical perspective, XML-based solutions will provide new opportunities to integrate disparate tools in a more systematic and scalable approach with long-term practical benefits for requirements engineering.

I hope the reader has gained valuable insight into the benefits, the limitations and the challenges of integrating COTS tools for requirements management.

Acknowledgements

I would like to thank Bob Schwanke for his detailed review of this paper and his feedback on the content.

References

BERENBACH B (May 2004) The Evaluation of Large, Complex, UML Analysis and Design Models, *Twenty Sixth International Conference on Software Engineering (ICSE 2004)*, Edinburgh, Scotland.

BERENBACH B (October 2004) Comparison of UML and Text Based Requirements Engineering, *Nineteenth Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA),* Van Couver, British Columbia.

BERENBACH B (2003) Evaluating the Quality of a UML Business Model, *Eleventh IEEE International Symposium on Requirements Engineering (RE'03),* Monterey Bay, Ca.

BORLAND SOFTWARE CORPORATION (2005) Caliber <http://www.borland.com/caliber/>.

GALACTIC SOLUTIONS GROUP (2005) <http://www.galacticsolutions.com/GalacticWhitePapers>.

I-LOGIX INC (2005) Rhapsody <http://www.ilogix.com/rhapsody/rhapsody.cfm>.

RATIONAL CORPORATION (2005) Requisite Pro <http://www.rational.com/>.

REQUIREMENTS ENGINEERING TOOL VENDORS AND FREEWARE SUPPLIERS (2005) <http://easyweb.easynet.co.uk/~iany/other/vendors.htm>.

TELELOGIC AB (2005) Doors <http://www.telelogic.com/products/doorsers/doors/>.

VOLERE (2005) Requirements Tools <http://www.volere.co.uk/tools.htm>.