ReqIF: Seamless Requirements Interchange Format between Business Partners

Christof Ebert[1], Michael Jastram[2]

[1]Vector Consulting Services

[2]Formal Mind GmbH, Universität Düsseldorf

The primary sources of project risks and product problems are poor, missing, or changing requirements. Often, the underlying root cause is insufficient collaboration between business partners. This article provides insight into how to effectively collaborate in requirements engineering. We describe the Requirements Interchange Format (ReqIF) standard and technologies for seamless requirements development and management. We look forward to hearing from both readers and prospective column authors about this and the technologies and tools you want to know more about.

DISTRIBUTED SOFTWARE ENGINEERING is the consequence of the relatively friction-free economic principles of the entire software industry. Software can be exchanged easily, and collaborative development is something most students already practice today at university. Often the supply chains aren't even limited to software development but might involve hardware suppliers, which must exchange interface specifications to adjust their firmware in line with operating systems or middleware.

Many companies' engagements in supplier relationships with distributed software engineering occur because of a lack of their own competencies—an effective work-split across architectural functions and building blocks and perceived cost differences. Most of these companies engage globally active sourcing companies for components, subsystems, and services to achieve the fastest ramp-up of their distributed sourcing targets. With time, they realize that savings are much smaller and problems are more difficult to cure than before. Needed changes in product features don't arrive in time from their suppliers, and quality issues and the risks of insufficiently implemented functionality increase with each additional supplier. Disillusioned, many companies pay high prices in their supplier management and still miss their own deadlines or face quality penalties in tight schedules. So what went wrong? Supplier-relationship management and collaborative development bear many challenges, specifically if external suppliers have shared responsibility on some features that architecturally overlap with system components done by a separate party.

From surveys across companies, we found that the divide-and-conquer approach is especially ineffective. Requirements specify needs and solutions. Data from the evaluated projects showed that only 52 percent of the originally allocated requirements appear in the final released version of the product [2, 1, 4]. We can identify several supplier-oriented challenges:

- overlooked requirements that affect distributed functionalities;
- inadequate supplier representation dur-

Published in IEEE Software, September/October 2012, Volume29, Number 5

ing system analyses;

- lack of communication beyond functional requirements;
- lack of requirement inspection before allocation to a supplier;
- failure to look at requirements across sourced components;
- representation of requirements in the form of designs that overrestrict the supplier;
- insufficient change management with all stakeholders along the product life cycle; and
- needed specifications not being shared or being shared inconsistently.

In this article, we will look to lessons learned from collaborative requirements engineering (RE) to effectively and successfully manage relationships across several business partners. Specifically, we will show how to effectively manage requirements interchange.

I. Observations

Working with companies around the world in industries such as automotive, information and communications technology, aerospace, medicine, industrial automation, and transportation, we realize that insufficient stakeholder management is not primarily due to "politics" or "insufficient communication," as engineers often argue. Often the original equipment manufacturer or product manager develop requirements and then partition and distribute them to their many business partners, such as suppliers. After the project is started, changes will become necessary somewhere along the way. Those are again partitioned and distributed to business partners. When the components finally arrive, integration is difficult and the product manager (or integrator) realizes that there was too much room for interpretation and that many engineering decisions were taken without synchronizing across the multiple stakeholders.

II. Requirements Exchange between Business Partners

Requirements exchange within an organization is rarely a problem, because most commercial RE tools use a central data repository. Because organizations typically use the same tool internally, this poses few problems. Things change when the requirements leave the organization. Business partners typically only want to reveal a subset of the requirements to outside parties, and might not want to open themselves directly to the Internet or other business partners.

Document-Based Solutions

A common practice is to transfer requirement specs in HTML, word processing documents, or spreadsheets between buyer and supplier [1]. These documents are primarily extracts from RE tools. Extracting more than one specification is generally not possible. Traceability is hard to support. The partners must agree on a template format so that partners subscribing to an exchange have a chance to import documents into their respective RE tools. Conflict detection and resolution are often not possible because most RE tools consider documents as new imports into the database or because a merge cannot be performed on a given view. A common alternative, namely PDF reports, makes it next to impossible for most RE tools to import the requirements because most structural information is lost and only the presentation of the data is preserved.

Other approaches merely extend existing tools with specific "bridges" to collaborate in specific environments and to ease copying and pasting of information from one tool to the other. For instance, major electronic

records management (ERM) vendors evolved their environments to support engineering document management. Customer-relationship-management (CRM) environments have since integrated with change and requirements tools. However, such IT-centric approaches don't go beyond interface management and don't integrate with systems and software engineering processes. Their scope is limited to interfaces and front-end processes.

More recently, Product Lifecycle Management (PLM) and Application Lifecycle Management (ALM) environments include requirements management components or allow federation with popular requirements tools. Such extended PLM and ALM solutions typically allow shared data models and a common data backbone; however, they still expect all distributed users to have the same tool.

Specific RE Tools

In general, all RE tools offer file-based capabilities for archiving and restoring specifications. An archive's file format is proprietary to a tool vendor and cannot be used if exchange partners use different RE tools. Moreover, archiving solutions aren't necessarily tailored to export a subset of requirements, so their usability reduces when each partner needs a separate view of the various subsets.

Some tool vendors do provide exchange features based on proprietary formats and are designed either to process a single specification at a time or to exchange specifications with a single partner. As we mentioned earlier, such exchange formats don't have wide industry acceptance. Furthermore, tool vendors accept the fact that customers will demand open standards for data exchange. Consider IBM's tool Rational DOORS. Initially, there was a commercial extension available called DOORS eXchange for exchanging requirements data between two parties. This extension has since disappeared from IBM's offering. Instead, ReqIF (the Requirements Interchange Format standard) is now advocated as support for data exchange. The high number of tool vendors who participate in the ProSTEP implementor forum[1] also indicates that the area of proprietary exchange solutions is diminishing.

Although there are differences across commercial tools in the marketplace, there are some core features that all significant commercial players offer:

Arbitrary attributes for requirements. In most tools, requirements have an ID and text, but users can add an arbitrary number of additional, typed attributes.

Hierarchical arrangements of requirements. Requirements are typically presented in a hierarchical structure. This presentation is often table-like, allowing users to see many or all attributes in columns. These arrangements typically correspond to a requirements document.

Traceability between requirements. Most tools can establish arbitrary traces between requirements, which can typically have attributes as well (like requirements).

A tool's value typically lies far beyond these data structures in the form of advanced reporting and collaboration features. These data structures matter, however, for achieving interoperability among different requirements tools.

Requirements Interchange Format

The requirements interchange format was started in 2004 as a global initiative in the automotive industry and became an Object Management Group standard in 2011 [3]. Today, it's used across many industries, making its way to domains such as transportation, industrial automation, and medical devices. Re-

[1] http://www.prostep.org/en/projects/internationalization-of-the-requirements-interchange-format-intrif.html

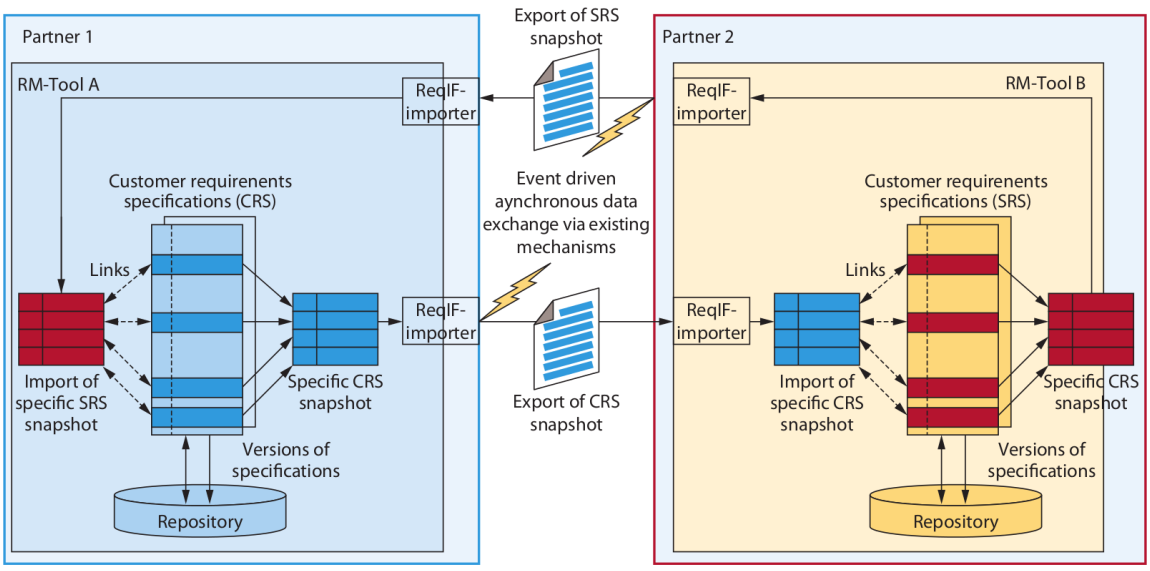

Figure 1: The exchange of requirements between two collaborating partners with different tools using ReqIF

qIF (formerly known as RIF) is based on XML and defines a tool-independent exchange format that is specifically defined to represent all important features of requirements data, including requirements in the form of attributed data elements, links (traces), views on the requirements data, and permission data. Additionally, text requirements can be exported together with multimedia content such as pictures or embedded documents. Containers are also available to transport complete or partial requirement hierarchies.

ReqIF and its practical implementation facilitate the single source concept of requirements across organizational boundaries thus ensuring consistency across different requirements artefacts, less defects from requirements to specifications to engineering work products, speed in exchanging information and collaborating on solutions, and cost reduction.

Many tools in the marketplace currently support the previous RIF 1.2 version of the standard. Several tool vendors plan on supporting this new ReqIF by the end of 2012.

Attributes. In general terms, a ReqIF model contains attributed requirements that are connected with attributed links. The requirements can be arbitrarily grouped into document-like constructs, and the features of all common RE tools can be mapped onto ReqIF without losing information. In fact, the ReqIF metamodel is so rich that established RE tools often can't represent all the features that the metamodel supports. Therefore, loss-free data exchange via ReqIF is feasible across tool and organization boundaries.

Structure. The ReqIF data model is persisted as XML, representing a tree structure. The root element contains a header, tool-specific extensions, and the actual requirements data. The requirements data consist of a number of relevant data elements:

- *SpecObjects* represent the actual requirements. SpecObjects themselves have no user data—the data is provided in the form of attributes.

- A *Specification* is a hierarchical structure referencing SpecObjects. Specifications represent what is typically considered the requirements document.
- *SpecRelations* represent directed links between SpecObjects. A SpecType is a collection of AttributeDefinitions that determine which attribute values the element can have. Supported atomic data types for AttributeDefinitions include numeric, string, enumeration, binary, and formatted text.

Note that SpecObjects, specifications, and SpecRelations can also have SpecTypes, and therefore attributes.

A Use Case

The ReqIF standard describes a number of use cases. Figure1 depicts the most prominent one and concerns the round-trip data exchange between manufacturer and supplier, and is depicted in Figure1.

In this use case, partner 1 exports a subset of requirements (with a subset of attributes) as ReqIF. Partner 2 then imports these requirements into his or her RE system. In that system, partner 2 can augment the requirements with additional attributes or add more requirements, even information that's relevant to only the supplier. It's possible to create links between requirements in the same or different specifications.

Once this work is done, partner 2 creates another ReqIF export. Again, this export only contains requirements and attributes that are relevant to partner 1, who in turn imports it into his or her system. Elements are matched by ID, and the new data that partner 2 provided are updated in the correct space.

This cycle can be repeated several times. Furthermore, the manufacturer can define a number of exports for various suppliers and merge them back into their database to have all suppliers' feedback listed side by side.

III. An Example

The Eclipse Requirements Modeling Framework (RMF) [6] is the first open source cleanroom implementation of ReqIF. RMF consists of a data core capable of reading, processing, and writing ReqIF, and a GUI called ProR[2], which allows interactivity with ReqIF models.

RMF is designed as a generic framework for requirements modeling and consists of an Eclipse Modeling Framework (EMF)-based implementation of the ReqIF core that supports persistence using the ReqIF XML schema. The core also supports older versions (RIF 1.1a and RIF 1.2).

The GUI for capturing requirements is called ProR (see Figure2). It operates directly on the ReqIF data model—an advantage over existing requirements tools, which require a transformation between ReqIF and the tool's data model. Not all tools support all ReqIF features; therefore, information might be lost in their processes. Because ProR uses ReqIF as the underlying data model, it's currently the only tool on the market that supports all ReqIF features.

IV. Conclusion

ReqIF was started as a global initiative and is growing fast. Today, it's used across industries, having initially started in automotive supplier networks and now growing to domains such as transportation, industrial automation, and medical devices. Using Eclipse as the platform for ReqIF with, for instance, ProR makes federation across tools possible, including environments such as Topcased [5] and PREEvision[3],

[2] http://eclipse.org/rmf/pror
[3] http://www.vector.com/PREEvision

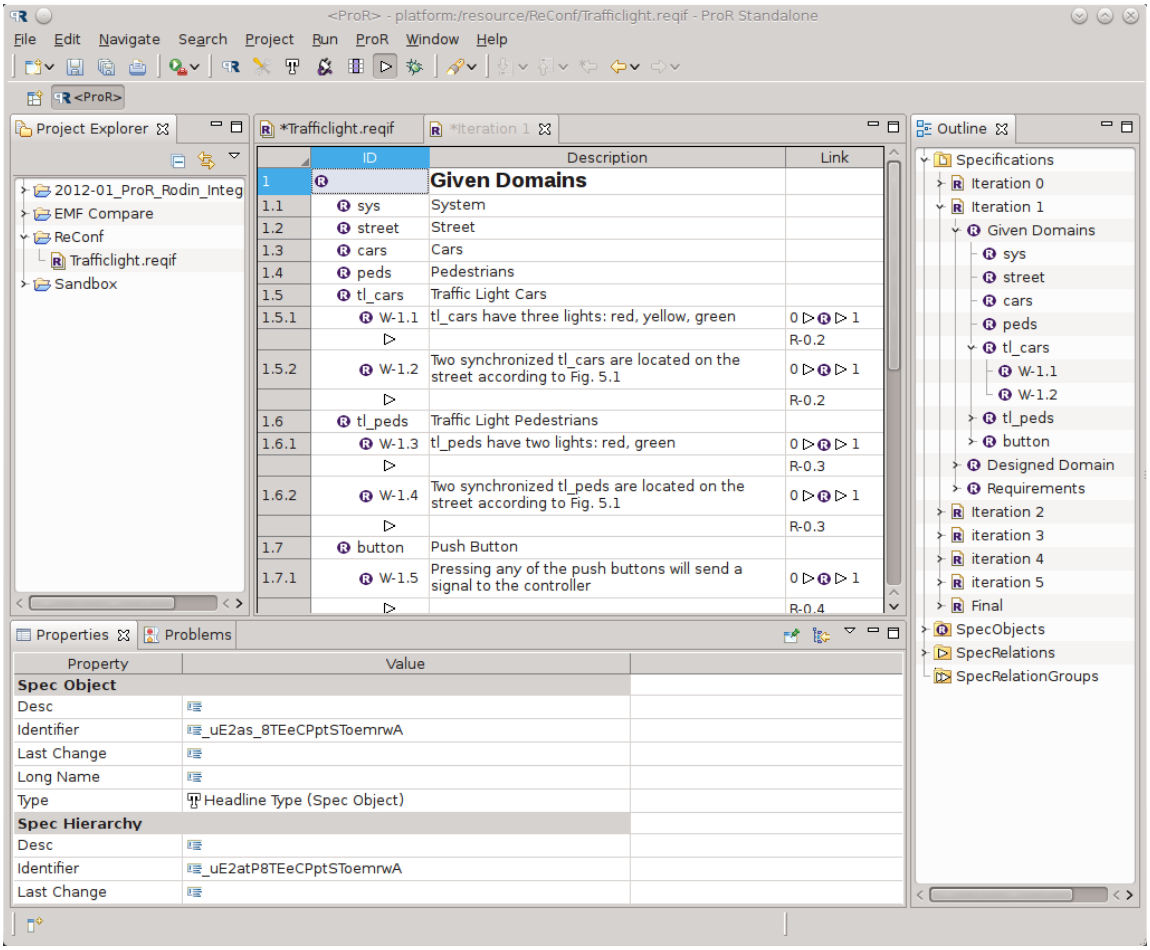

Figure 2: The ProR GUI is designed as an extensible application to the Requirements Modeling Framework.

thus allowing the seamless connection of requirements with architecture models along the entire product life cycle.

Needless to say that ReqIF by itself will not solve problems entrenched in insufficient engineering and management processes. Therefore a typical introduction strategy will always start with analyzing and improving processes and improving product life-cycle management – specifically the supplier interfaces – and then proceed to using ReqIF for streamlining and automating the collaboration in requirements engineering and management from concept to production and maintenance.

Our results from ReqIF and its predecessor showed cost and resource savings in the exchange between customers and suppliers. These cost reductions resulted from improved efficiency (that is, less manual exchange many with small adoptions) and from less rework due to changes and misunderstanding by sharing insufficient requirements information. This holds true specifically in time-critical stress situations, such as changes in the final phase of an

RFP, change requests close to product delivery, or at the start of production. When such requirements changes must be quick, they often create many defects and inconsistencies. Such expensive last-minute defects are reduced by exchanging traces between customer requirements and proposals. Therefore, such collaboration not only makes supplier management more effective and less error prone, but also tangibly and sustainably reduces cost along the product life cycle—which, these days, is certainly important in global competition.

References

[1] C. Ebert. Global Software and IT. Wiley-IEEE CS, 2012.

[2] C. Ebert and R. Dumke. Software Measurement. Springer, 2007.

[3] Object Management Group. Requirements Interchange Format (ReqIF), version 1.0.1. http://www.omg.org/spec/ReqIF/1.0.1.

[4] Standish Group. What Are Your Requirements. Tech Report, 2003.

[5] M. Jastram and A. Graf. Requirement Traceability in Topcased with the Requirements Interchange Format (RIF/ReqIF). 1st Topcased Days Toulouse, 2011.

[6] M. Jastram and A. Graf. ReqIF -– The New Requirements Standard and its Open Source Implementation Eclipse RMF. Commercial Vehicle Technology Symposium, 2012.