

Improving Requirements Engineering within the European Space Industry

Silvia Mazzini, John Favaro, Rudolf Schreiner, Ulrich Lang, H-P de Koning

▶ **To cite this version:**

Silvia Mazzini, John Favaro, Rudolf Schreiner, Ulrich Lang, H-P de Koning. Improving Requirements Engineering within the European Space Industry. Embedded Real Time Software and Systems (ERTS2012), Feb 2012, Toulouse, France. hal-02263379

HAL Id: hal-02263379
https://hal.science/hal-02263379

Submitted on 4 Aug 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Improving Requirements Engineering within the European Space Industry

Silvia Mazzini, John Favaro
INTECS
Via Umberto Forti Trav. A, n.5
Polo di Attività Montacchiello, Loc. Ospedaletto
I-56121 Pisa, Italy
Silvia.Mazzini@intecs.it

Rudolf Schreiner, Ulrich Lang
ObjectSecurity Ltd.
St John's Innovation Centre
Cowley Road
Cambridge CB4 0WS
United Kingdom
ras@objectsecurity.com

H-P de Koning
European Space Agency
Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands
Hans-Peter.de.Koning@esa.int

Congress Topics: Process, Methods and Tools: Requirement Engineering, Languages, Model-based System Engineering

Domain: Space

The Next Generation Requirements Engineering (NextGenRE) (ESA/ESTEC Contract 4000101353/10/NL/SFe) project seeks to identify possibilities to improve the requirements engineering process within the European Space industry in connection with Model-based System Engineering (MBSE).

Questionnaire on current and future practices

The preliminary activity of the project included studying the requirements engineering process and workflow within the Space industry and more broadly within the community of systems engineering organisations. To this purpose a questionnaire was developed, in which the primary salient elements of requirements engineering in the target organisations were investigated. The questionnaire was sent out in the beginning of 2011 to approximately 200 recipients in a number of different organisations. The response rate was an extremely high 50%, due to the targeted nature of the distribution list, which was able to identify motivated and enthusiastic individuals to contribute feedback from their experience. The main questionnaire results are summarized in the following discussion.

Current practices

Among the respondents to the questionnaire, by far the largest sector was Aerospace / Space, followed by a significant portion from the automotive industry. A broad geographical distribution was also revealed, including overseas (e.g. NASA/JPL in the United States). According to the respondents, requirements engineering is mostly carried out by internal engineering personnel, but in many cases requirements are provided directly by customers – evidence of the customer/supplier chain in the Space industry.

The standard requirements engineering process is split evenly between ECSS (Space) and DO178 (Aerospace) – however, a significant portion of respondents noted that they use non-standard, organisation-specific processes. Certification, when it exists, is generally ISO900x. Other types of quality certification such as CMMI and SPICE remain relatively unused in the Space industry.

Requirements engineering in the Space industry is dominated by the use of DOORS and Office tools (Word and Excel). Requirements exchange is mostly through Office and DOORS files – but with a significant continuing use of paper, related to IPR protection and security concerns (Figure 1). Little use of open exchange formats was reported.

Of particular interest was the perceived importance of support for distributed requirements engineering. Few respondents reported current engagement in distributed requirement engineering, probably because they also felt that

the current level of tool support is sorely inadequate. Once again, security and IPR protection issues were highlighted by respondents as problems.

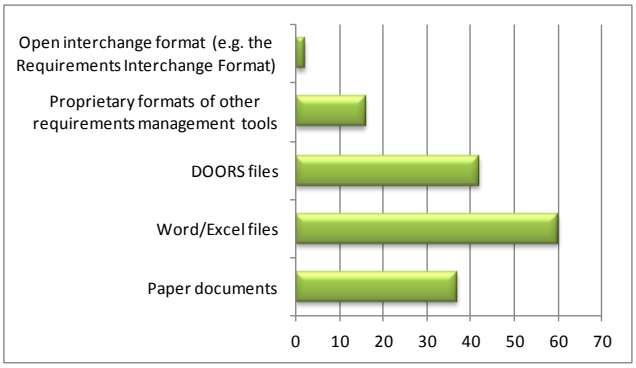


Figure 1: "How do you exchange requirements with external environments?"

The automatic generation of requirements documentation was considered by all respondents to be important, but in many cases was perceived to be so inadequate that automatic documentation generation was not practiced.

Guidelines and glossaries are in significant use, with some (lesser) use of templates reported – and a considerable portion of respondents still currently write requirements in free form. The use of non-textual forms for requirements (e.g. explicit mathematical expressions or tabulated parameter / value sets) appears to be a growing phenomenon, but the amount of tool support currently provided is perceived to be only partial or, in many cases, totally absent. Some respondents noted that such forms were mostly supported by engineering tools such as Matlab, whereby a major problem was the lack of a proper underlying semantic data model.

Requirements reuse is an important issue in requirements engineering and represents an important opportunity, but many respondents felt that tool support remains highly inadequate. Indeed, most stated that their current technique for requirements reuse was simple cut-and-paste using Office tools.

The principal means of requirements verification in most organisations – especially those directly involved in the Space sector – consists of structured reviews and inspections. As yet, little activity in formal or even semi-formal verification was reported, due to problems of tool support and access restrictions.

Future practices

The context for a discussion of desired future practices was set in the questionnaire by a survey of the amount of model based development currently practised in the organisation. Most model based development is based on Matlab/Simulink with engineering models; however, a growing number of respondents reported on the use of the more general modelling languages such as UML and (to a lesser extent) SysML [OMG 2011a]. Respondents associated a high level of importance with model based development, and in some cases reported good tool support – presumably through the commercial engineering tools (Matlab family).

Respondents attached a high level of importance to support for automated conflict analysis and automated analysis of traceability from higher to lower levels. A split of opinions regarding automated assistance for requirements quality analysis was perceived: as many respondents were enthusiastic as neutral. Interest in support for executable requirements was decidedly low, with only a minority expressing a positive opinion. When the subject turned to advanced facilities for assisted verification, however, the interest was firmly positive.

A high level of interest in the possibility of advanced facilities for discussion and negotiation of requirements was expressed by respondents. Many respondents expressed great interest in facilities for managing multiple views on requirements, with some of them considering such facilities to be of critical importance (Figure 2).

Enthusiasm for the possible integration of applicable standards (e.g. ECSS) directly into requirements management facilities was extremely high, due to the possibility of making them directly available for tailoring and referencing. Although many respondents expressed strong interest in the integration of glossaries (e.g. through hyperlink facilities), an equal number expressed perplexity and unfamiliarity with this approach.

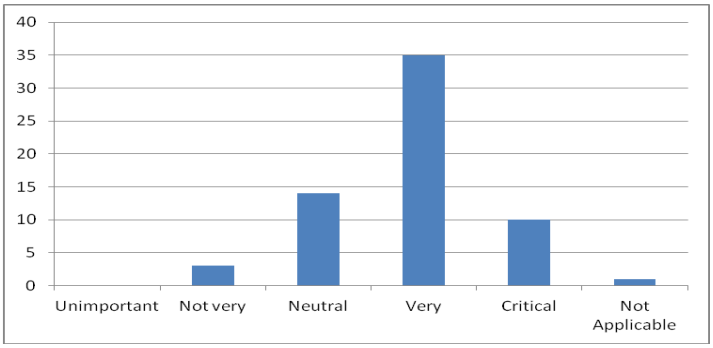


Figure 2: "How important would it be to support multiple views on requirements?"

Final respondent comments

When asked to name the primary deficiencies in their current requirements management practices, some lamented a lack of Agency-wide rules, resulting in too personalised styles. Many complained that the rationales behind requirements are too often not recorded. When asked about desired improvements, several respondents mentioned an agreed, non-proprietary exchange format for requirements (Figure 3).

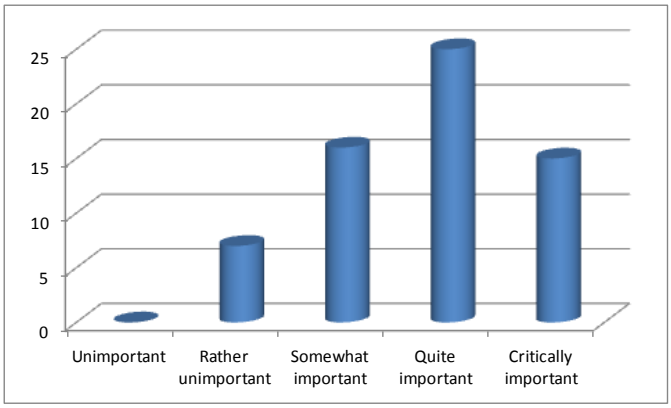


Figure 3: "How important would it be to have an open requirements interchange standard?"

Others expressed interest in the introduction of ontology and knowledge based facilities. Still others associated high importance with facilities to manage the current extreme complexity of requirements management.

At the end of the questionnaire, respondents were invited to comment on whatever aspects of requirements engineering they felt to be important to bring into the discussion. Some respondents emphasized the role of requirements engineering as a means for proper communication among stakeholders. Others reiterated the complexity and unfriendliness of current tools.

The NextGenRE approach

Based on the results of the questionnaire, the project has focused on the development of a tool supporting advanced requirement engineering capabilities for practical short and mid term improvement of MBSE. The principal elements of this approach include:

- Addressing the inherent ambiguity of natural language requirements expression, trying to “computerize” this human process and knowledge as much as possible, in order to support the reuse of *a priori* knowledge of requirements and design, best practices and lessons learned. This is done by application of the following main concepts:
    - Ontologies and Models: use of (semi) formal methods to describe systems and their properties, mainly using models and ontologies.
    - Semantic glossaries: exact descriptions of terms in the form of glossaries with semantic relationships between them.

    - Templates/Patterns: use of predefined textual templates, in many cases coupled to expressions stored in the formal model, as a means for promoting requirements reuse.

- Integration of Requirements Engineering into Knowledge Management. Requirements are considered as part of the integral knowledge about a system;

- Enabling full integration into a well-defined MBSE process, including the integration of Requirements Engineering and Management (REM) tools into MBSE tools;

- Establishing a sound and practical basis for the exchange of requirements between tools and the external environment.

The Requirement Engineering and Management Environment

We have implemented semantic support for a powerful open source wiki [XWIKI 2011]. This semantic wiki environment allows the definition of semantically augmented requirements based on templates, baselines, previous projects or preliminary design information. This enables the use of technologies and tools from both the ontologies and the modelling worlds on a complete knowledge base of the system, containing requirements and design information, e.g. to analyze the system, to automatically generate products like documentation using model transformations or to support traceability over the whole system life cycle.

A typical usage scenario

The main concepts behind the use of the NextGenRE tool for Requirements Engineering and Management (and for knowledge management in general) are the abilities to define semantic properties for requirements represented as wiki pages, to establish any kind of semantic relationship between pages, to use advanced query capabilities to search for specific pages, to generate pages from templates, and to use queries in selecting arguments during the template based page generation process. Below, the principles of operation are outlined through a typical scenario.

The System Engineer (SE) starts with the selection of the mission type, e.g. telecommunications or earth observation. From the corresponding template, the highest level requirement page is generated. During the page generation process, the wizard asks for a set of relevant information at the current level of abstraction. This includes generic requirements engineering information, e.g. the owner of the requirement, and also specific information for the current requirement. What exactly the wizard requests for a specific requirement is defined in the page template and the underlying semantic database. For example, in high level mission requirements, the wizard might ask about the orbit of a spacecraft. The generated pages include semantic links to all relevant pages, e.g. the high level mission page includes links to all pages representing the associated requirements and to the next lower level of decomposition. These links are not hardcoded in the pages, but defined as SPARQL [W3C 2011] queries which are executed in page view. The user is able to navigate the pages. It is just necessary to click on the selected result of the SPARQL query to navigate to this result.

At all times it is possible to modify pages, including their textual content, semantic information expressed as macros in the page (both properties and queries), object properties associated with a page, and semantic information associated with these page properties. For example, the software engineer can modify a requirements text and then change the state of a requirement from “undefined” to “defined”. The status of the requirement is implemented as an object property linked to a semantic property, so in order to change the status, the engineer has to use the object editor to modify the status, and the semantic property is automatically updated as well. The update happens during the page save action. When any other page is viewed, then the update is immediately reflected. For example, a page describing an architectural element might include SPARQL queries to list all “undefined” and all “defined” requirements associated with this element. When the status of a requirement is changed, the related requirement automatically moves from “undefined” to “defined”. For using semantics in the requirement content (the “string” of current REM systems) it is possible to add arbitrary semantic macros to the requirement content, e.g. setting a property, obtaining a property or a query.

The engineer now continues with iterating the processes of architectural decomposition and establishment of the requirements from templates. These templates might be very specific and well defined, giving a semantically very strong definition, but there is also a “SimpleRequirement” template for simple texts. This simple requirement text then can modified as required by the engineer to express her exact requirement in a flexible way, not being pressed into the mould of something strongly fixed.

Using tick-boxes and selections, it is possible to define arbitrary requirements at all times, e.g. “derivedfrom”, “inherited”, “satisfied” and so on. The details of the exact requirement format are determined by domain experts, and are generally adapted to current practice and standards like ReqIF, to provide a high level of compatibility.

Multiple views through multiple environments

As noted in the questionnaire responses, there is a high degree of importance associated with the provision of multiple views onto requirements. Implicitly, however, a single "requirements engineering and management environment" is assumed by the respondents. NextGenRE embodies an approach where the provision of multiple views is accompanied by the provision of multiple environments, thus expanding also the notion of a view to include questions of where requirements engineering and management is best done.

A key aspect of the scenario described in the previous section is the concurrent activities of requirement engineering and architectural decomposition, corresponding to one of the main premises of the NextGenRE approach. This is supported in the tool through dual environments that effectively provide the user with dual views on the requirements engineering process. Furthermore, there is an important third environment that is explicitly assumed: the *external* environment. That is, the exchange of requirements with external entities is elevated to first-class citizenship within the NextGenRE environment; this is due to the importance both of distributed requirements engineering and of the need to be able to exchange information with other requirements engineering tools (Figure 4).

As a result of this set of 2+1 environments, we arrive at one of the central challenges of requirements engineering, the interpretation and management of requirements across environment boundaries. This central challenge is already being addressed by the OMG Requirements Interchange Format (ReqIF) [OMG 2011b], concerning the interpretation of requirements across external boundaries. But in addition, we are also addressing this challenge with regard to *internal* boundaries between different requirements views.

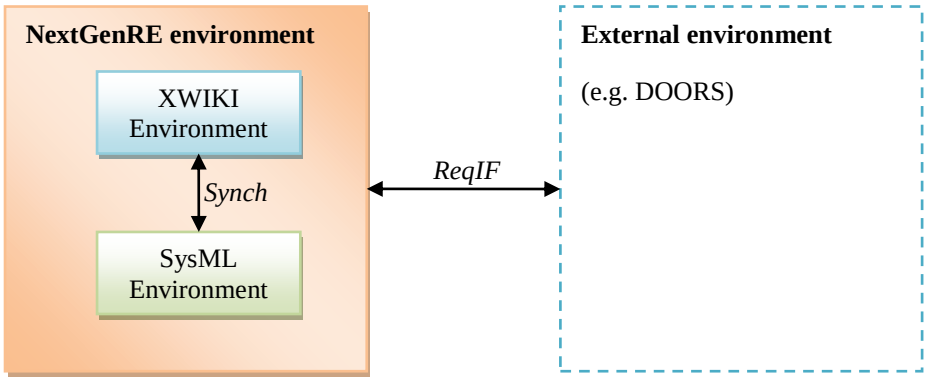


Figure 4: Dual requirements engineering environments and explicit external environment

The NextGenRE semantic wiki environment – requirements import and edit

The semantic wiki is the environment made available for requirements creation and editing. Wiki environments are naturally suited to straightforward entry and sophisticated editing of text (as well as offering a natural basis for collaborative work), and it is not surprising that other work has been done in connection with the use of wikis for requirements engineering [GEI 2009].

We use this environment as a point of departure to examine the first matter of cross-environment interpretation: importing requirements from an external source. As the questionnaire responses indicated, current users in the Space industry place a high degree of importance on the introduction of a non-proprietary requirements interchange format, and we have implemented a ReqIF capability for this purpose. The ReqIF provides the basic mechanisms for requirements interchange, but many decisions must be taken by both user and implementer, as the following walk through the import process will illustrate.

The import dialogue (Figure 5) begins the process. A ReqIF file is selected and parsed, identifying the various attributes and values of the requirements as provided in the ReqIF. Here the first interpretation and decision-making processes take place. The NextGenRE approach involves the mapping of each individual requirement to its own, individual wiki page. This requires a decision by the user as to which attribute to use for naming the wiki pages (since no such concept exists in the external environment); and it may require adjustments, since fields may contain duplicates and cause wiki page overwrites and similar side effects.

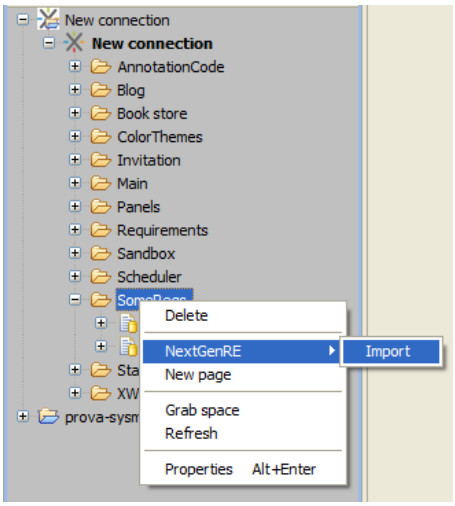


Figure 5: Importing requirements with the ReqIF

A key characteristic of the ReqIF is its ability to conserve hierarchical structure among the requirements. Within the wiki environment, this structure is preserved and can be displayed in the XEclipse explorer bar (Figure 6).

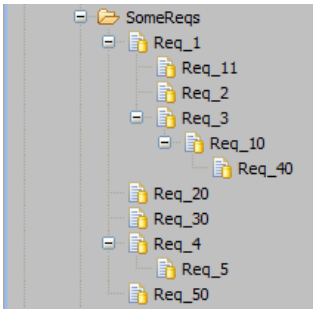


Figure 6: Conserving hierarchical requirements structure during import

But how can rich, hierarchical structure be managed in the conceptually flat, hyperlinked wiki environment? This is where the features of the NextGenRE *semantic* wiki come into play. The incoming requirements, with their hierarchical and attribute structure, are not merely mapped onto flat wiki pages, but also have the associated semantic information (e.g. parent and child relationships) stored in the NextGenRE semantic database in the form of standard RDF triples, the representation of choice in the Semantic Web. For example, in the extract from the database shown in Figure 7, the triple represented by the third line records the fact that Requirement 1 is a parent of Requirement 2. Likewise for attributes: for example, the first line records the fact that the ID of Requirement 11 is the string "Req. 11".

subj text	prop text	obj text
SomeReqs.Req_11	ID	Req.11
SomeReqs.Req_2	Description	Requisite number 2 (sub-req of 1)
SomeReqs.Req_2	http://www.nextgenre.org/req/parent	SomeReqs.Req_1
SomeReqs.Req_2	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://automotive-his.de/200807/rif/ef#SpecObject
SomeReqs.Req_2	ID	Req.2
SomeReqs.Req_20	Description	Requisite number 20
SomeReqs.Req_20	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://automotive-his.de/200807/rif/ef#SpecObject
SomeReqs.Req_20	ID	Req.20

Figure 7: RDF storage of semantic requirement features

Thus, the full hierarchical structure of the incoming requirements is preserved in a semantic database, as well as all attributes. The power of this approach becomes evident when querying the database, using the SPARQL query language. All of the information associated with the requirements set can be accessed, retrieved, and presented dynamically through the same, familiar wiki interface. For example, a set of SPARQL queries could be issued to show the parent (if any), other children (if any), the attributes of the requirement associated with the page and all root requirements, i.e. those requirements that don't have a parent. Figure 8 shows the XWIKI page associated with Req. 3 from Figure 6.

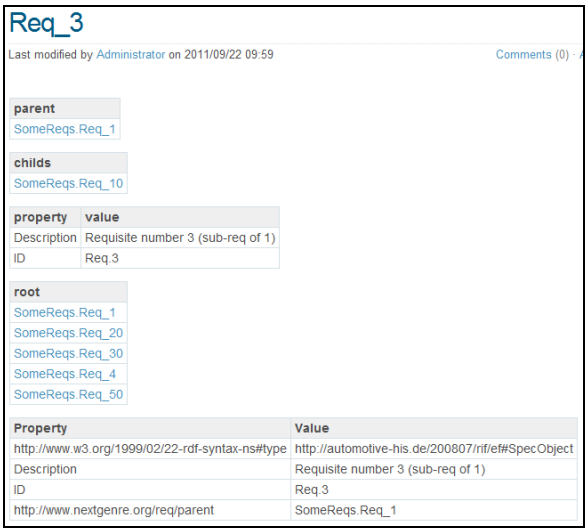


Figure 8: Dynamic query-based wiki display of requirement with structure and attribute information

The user's decision making process isn't finished with the indication of which attributes to use to identify wiki pages. The user must also decide which attributes to import in the ReqIF file. He may choose to import all attributes available in the ReqIF specification, or restrict the import to a specified subset. Once specified, this subset of attributes is then mapped to the wiki page reserved for the requirement.

The end result of this process is that the user is provided with a controlled import process, whereby the information coming in from the ReqIF import file structure is guided by user decision-making to be set up in a wiki environment without sacrificing semantic richness (structure and attributes). The semantic database underlying the wiki ensures that the full specification of a requirement in the ReqIF is preserved and can be manipulated. The native wiki capabilities ensure a powerful requirement editing and management environment.

The SysML environment – relating requirements to architecture

The Papyrus SysML editing environment on Eclipse [PAP 2011] is exploited for the other pillar of the NextGenRE approach: relating requirements directly to architectural building blocks. Some of the most interesting aspects of dealing with requirements views arise in the management of the dual environment. When a single environment is either explicitly or implicitly assumed, then all requirements engineering activities are obviously carried out in that single environment. But when *two* environments are provided, then a choice can (and must) be made – and then enforced – concerning which activities are best performed in which environment.

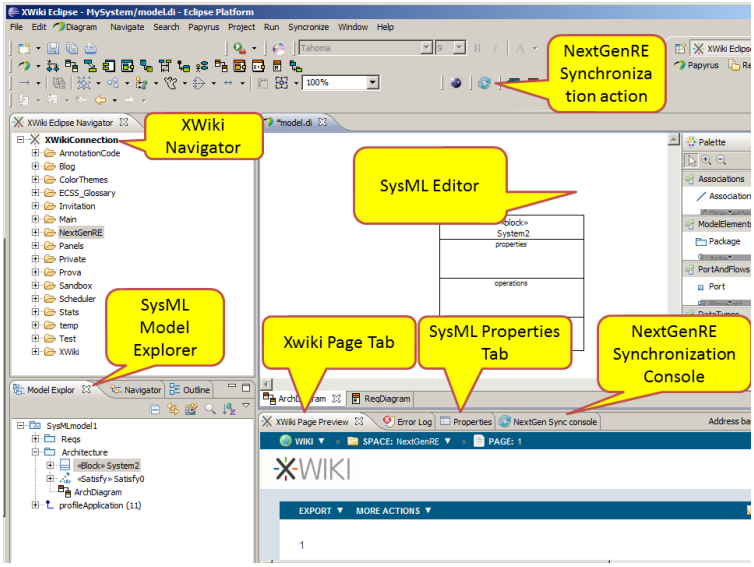


Figure 9: Dual requirements engineering environments in NextGenRE

NextGenRE Eclipse plug-ins support XWiki (using XEclipse in the Eclipse environment) and SysML tool and model integration. In particular it is possible to automatically manage XWiki pages as stereotyped requirements in a given

SysML model. Figure 9 shows an example of how Eclipse views can be configured in order to enable working within the dual environments. At the left, the two navigators are depicted: above, the wiki environment navigator, and below, the SysML model explorer.

Integration between XWiki and a given Papyrus SysML model/diagram is provided through the NextGenRE *SysML Gateway*. After the SysML Gateway has been activated it is possible to drag a requirement from the XWiki Eclipse Navigator to the Papyrus SysML Editor – thus crossing an internal border between two different environments (Figure 10).

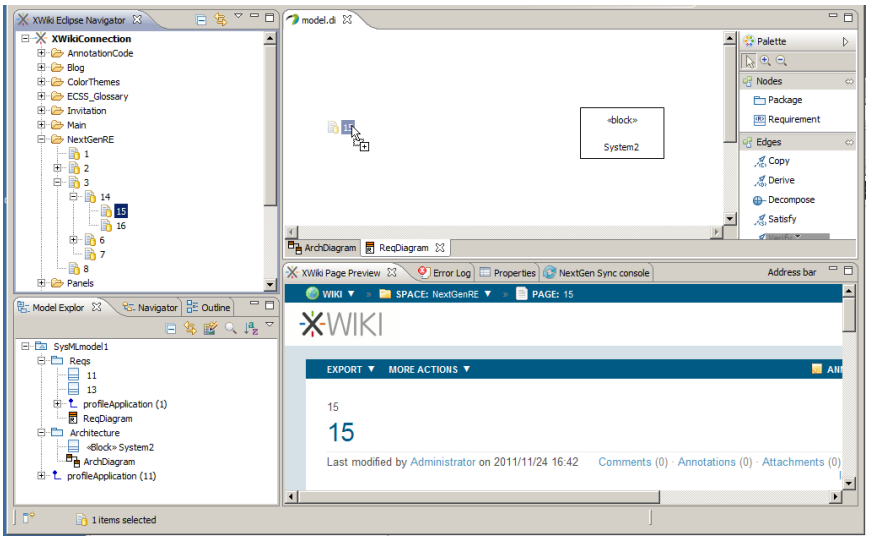


Figure 10: Dragging and dropping a requirement across environments

When the page is dropped into the SysML requirement diagram the tool creates a SysML requirement both on the graphical editor and on the SysML model underneath. The SysML requirement is automatically stereotyped as a NextGenRequirement; in order to accomplish this, the NextGenRE profile is automatically applied to the SysML model, if not yet applied (Figure 11).

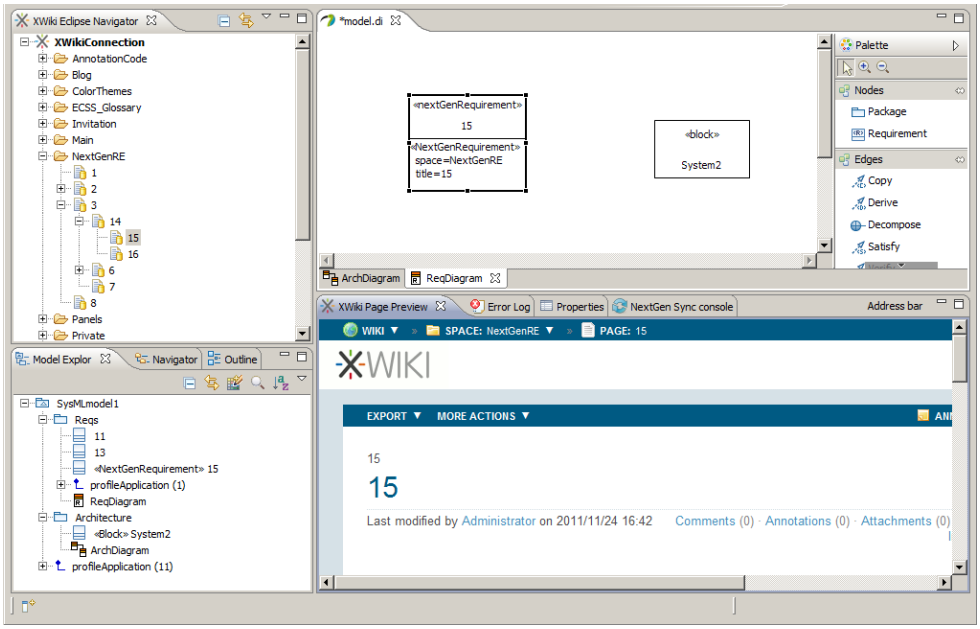


Figure 11: Creating a NextGenRE SysML requirement and freezing its contents

The NextGenRequirement stereotype extends the SysML Requirement construct by adding XWiki specific information (page's space and title). More importantly, the NextGenRequirement stereotype comes with constraints stating that SysML requirement information such as text is not relevant in this environment (and thus "frozen"); this is part of the NextGenRE approach which foresees that the content of requirements must be viewed and edited in the XWiki space, the more natural environment for requirement modelling, leaving SysML as the proper space where system architecture

and so traceability from requirements to architecture must be modelled. Following this approach the modeller, in order to check the requirement properties, does not use the SysML Property editor, but rather must refer to the corresponding XWiki page.

According to the NextGenApproach the SysML architect can model *satisfy* relationships from architectural elements to NextGenRequirements, by using standard SysML Satisfy relationships; these relationships are available to the modeller in SysML Requirement diagrams where also architectural blocks can be visualized (by drag and drop) from the SysML Model Explorer view.

When the SysML model is saved all the modifications performed upon NextGenRequirements, more properly upon related *satisfy* relationships, are reflected in the corresponding XWiki pages (Figure 12).

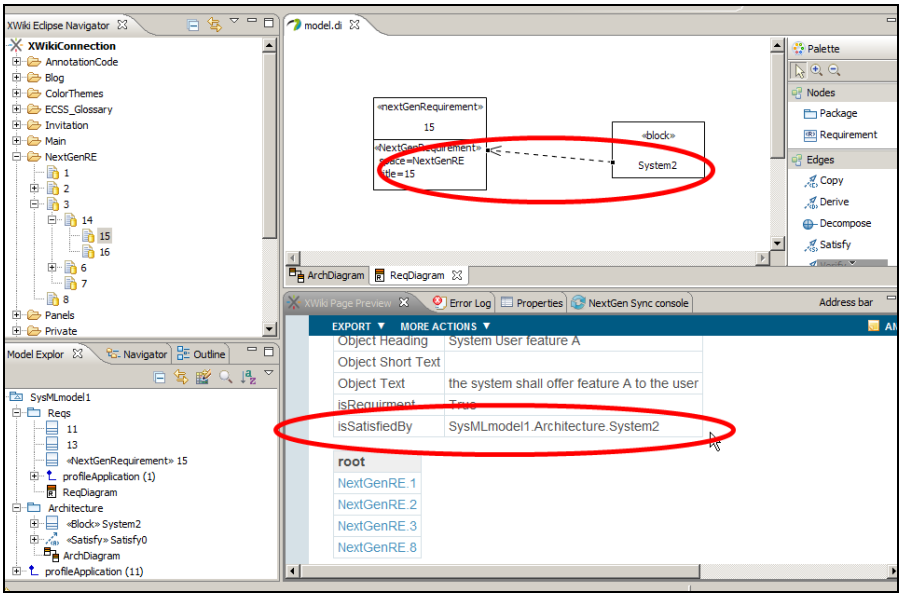


Figure 12: Mapping of SysML satisfy relationship back into wiki environment

In particular, for each satisfy relationship having NextGenRequirement ReqA as target and a SysML element B as source, a triple is automatically maintained in the XWiki page associated to ReqA, reflecting the link to the architectural element.

Synchronization between the two environments can be a challenge. For example, it is possible that a user will delete a requirement in the SysML environment, which must be correspondingly deleted in the wiki environment. A number of strategies have been elaborated both for automatic and manual synchronization between the two environments to keep the information consistent across them.

Exporting requirements with the ReqIF

The final context in which requirements traverse environment borders is the export of requirements to the external environment. Again taking the cue from the results of the questionnaire, we have concentrated on the external DOORS environment as an important source and sink of space engineering requirements.

NextGenRe allows the user to select a wiki space from which to extract all requirements to be saved to a ReqIF file. The export process maps the semantic attributes/fields of each page (corresponding to a requirement) into the output ReqIF file and adds some information required to make it compatible with DOORS 9.3. NextGenRE pre-defines the same DOORS types and attributes to make the generated RIF file as compatible as possible to DOORS. Attributes not defined by DOORS and added by user are added by NextGenRE as String attributes. This also requires an explicit decision process by the user: given the very nature of NextGenRE as an advanced requirements engineering environment, it is inevitable that semantic information and capabilities will be added that are not supported in environments such as DOORS. An important part of the project and its future extensions will be the study of appropriate mappings from NextGenRE to the external environment so that as much as possible is preserved in the most reasonable manner.

Currently DOORS itself is making the transition from earlier versions of the ReqIF to the latest version supported by the OMG. We are keeping NextGenRE closely aligned with the DOORS-supported versions, while preserving upward compatibility with the upcoming versions of the ReqIF.

References

[GEI 2009] Michael Geisser, Hans-Jörg Happel, Tobias Hildenbrand, Axel Korthaus, Stefan Seedorf: "New Applications for Wikis in Software Engineering." PRIMIUM 2009: 145-160.

[OMG 2011a] Object Modeling Group, Systems Modeling Language (OMG SysML™), SysML v1.2 Specification.

[OMG 2011b] Object Modeling Group, Requirements Interchange Format (ReqIF), ReqIF v1.0.1 Specification, April 2011.

[PAP 2011] Papyrus Eclipse Project, http://www.eclipse.org/modeling/mdt/papyrus/

[W3C 2011] The SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/

[XWIKI 2011] www.xwiki.org