Next Generation Requirements Engineering

John Favaro
Silvia Mazzini
Intecs Spa
Via Egidio Giannessi 5
Loc. Montacchiello
56127 Pisa – ITALY
John.Favaro@intecs.it
Silvia.Mazzini@intecs.it

Hans-Peter de Koning
European Space Agency / ESTEC
Keplerlaan 1
2201 AZ Noordwijk
The Netherlands
Hans-Peter.de.Koning@esa.int

Rudolf Schreiner
ObjectSecurity Ltd.
St John's Innovation Centre
Cowley Road
Cambridge CB4 0WS
United Kingdom
rudolf.schreiner@objectsecurity.com

Xavier Olive
Thales Alenia Space France
26 Avenue Jean François Champollion
31100 Toulouse - FRANCE
Xavier.Olive@thalesaleniaspace.com

Abstract. Establishing and managing a "good" set of requirements is one of the critical success factors for any space system project, and for the development of any complex product in general. The NextGenRE project sponsored by the European Space Agency has developed an innovative approach featuring the use of semantic wiki technology to enable users to specify structured, semantically-rich requirements associated with canonical system designs in OMG SysML. Reuse of requirements through templates is supported, as well as semantic reasoning to check properties such as consistency and coherence among requirements. The approach is entirely model-based, allowing tight integration of the requirements engineering and design processes and opening the way to the application of model checking and simulation techniques. The new approach is being validated alongside an existing space project.

Current Requirements Engineering Practice

It has often been said, with only slight exaggeration, that systems engineering is mostly about eliciting requirements and determining feasibility. Requirements engineering is particularly crucial in complex systems developments such as space systems, which need multi-disciplinary concurrent engineering of many different subsystems in domains ranging from electronics to biology. The Next Generation Requirements Engineering (NextGenRE) project sponsored by the European Space Agency has developed an approach to introducing incremental improvements in requirements engineering as it is practiced today toward the best practices of tomorrow.

It is important to distinguish between:

- *Requirements engineering*, which comprises the elicitation, formulation, analysis, derivation and refinement of requirements, as well as the identification and assignment of verification and validation approaches. Requirements engineering is usually performed in an iterative process with conceptual / preliminary design

activities.

- *Requirements management*, which consists of support for multi-user access to and configuration/version control of requirement sets, change impact analysis, change control, traceability between requirements at different levels and to design definition and verification and validation, as well as production of derived artefacts such as requirements traceability matrices and verification control documents. Requirements management is mainly performed after the initial requirements baseline has been established. In space projects this typically happens at the System Requirements Review halfway the preliminary design phase ("Phase B").

Survey. At the beginning of the project, a survey of current practices was made. A questionnaire was distributed amongst practitioners in aerospace and other industry sectors. Approximately 100 questionnaires were returned which was an excellent response (50%). Based on this survey and drawing from its own experience in space system engineering, project partner Thales Alenia Space identified a number of areas that need improvement in requirements engineering.

Tools and formats. Requirements are currently defined and stored in either dedicated requirements management tools/databases (primarily IBM Rational DOORS) or general purpose office documents (primarily Microsoft Word or Excel).

Data exchange. In most space system projects, some form of electronic data exchange (intra-company or with external companies) is mandatory. Office documents (Word, Excel) can obviously be used but quickly create configuration control and granularity problems. DOORS modules can be exchanged when different parties use the same version of the tool, but often this is not the case. On a large programme with many organisations involved, it is important to have easy-to-use and reliable interfaces to exchange sets of requirements between organisations. This is a problem because all organisations in one project usually cannot standardize on one version of one tool, while still needing bi-directional exchange. Improvements are expected from using a standardized data exchange format.

Collaborative requirements engineering. In the early phases of the system life cycle collaborative work precedes formalized data exchange: engineers are sharing requirements. This is particularly true in the engineering of complex systems such as those in the space industry where experts from many different disciplines practice concurrent engineering, often with the support of dedicated facilities such as the Concurrent Design Facility (CDF) at the European Space Agency and similar concurrent design centres in industry. It is common for requirements arriving from the various disciplines to interact, and even compete, with requirements from other disciplines. A typical example is in the area of "budgets", where a common resource such as communications bandwidth or electric power must be shared among the different subsystems. Thus it is extremely important that the collaborative work environment render as open and as transparent as possible the requirements coming from the disciplines so that the process of reconciling and balancing needs can be maximally effective. The supporting tools need to be easy-to-use and flexible.

Lacking support for traceability. Requirements engineering and support activities (verification, justification, etc.) lack strong links between them in practice. The main reason appears to be a tooling issue: different tools are used and the gateways between them are missing. The direct consequence is the difficulty of making the link between justification items or verification elements and the set of requirements. There is a need to improve the connection between the requirement engineering work on one side and the design and verification work on the other side. A great deal of human expertise is present in these activities and ways need to be found to connect requirements to justification and verification

in an efficient manner. The requirement-justification relation is certainly one of the most missing relations in current requirements engineering practice.

Allocation of requirements to design artefacts. Models are more and more used in the development process. These models are not sufficiently linked today with the requirement engineering / management tools. Some improvements could be introduced by having a tool (or a set of tools) that makes it possible to define the traceability from the user level requirement to the "final" model element (e.g. in UML, SysML) that satisfies this requirement.

Scalability of the requirements management process. A common problem has been encountered in all domains: having tools to properly manage the enormous amount of data associated to a large programme. Indeed initially only user level and technical requirements were present in the database. Now the scope is growing with the inclusion of the architectural design definition, customer requirements from the Statement of Work, and eventually about 20,000 requirements from the ECSS (European Cooperation on Space Standardization) standards – all with the aim of improving compliance and enabling precise verification and validation. Furthermore, these requirements are not always handled in the same way and with the same process, leading to a need to separate the requirements by category and to manage them depending on this categorization. Some light process for "non-technical" requirements should be defined, apart from the classical (more involved) technical requirements engineering process.

Reuse of requirements. In the space industry, it is common to develop and maintain *reference architectures*, often in the context of product lines. Dealing with reference architectures and related methodologies implies the reuse of requirements and the notion of *generic requirements*. Three cases can be considered when a new development starts (Figure 1):

- **Project A**: Reuse of a previous program (and product line) as-is (version V1);
- **Project B**: There are some commonalities between products, but not enough to reuse the current version. The specifications should evolve (version V2);
- **Project C**: There is a need to customize the requirements – that is, reuse with adaptation.

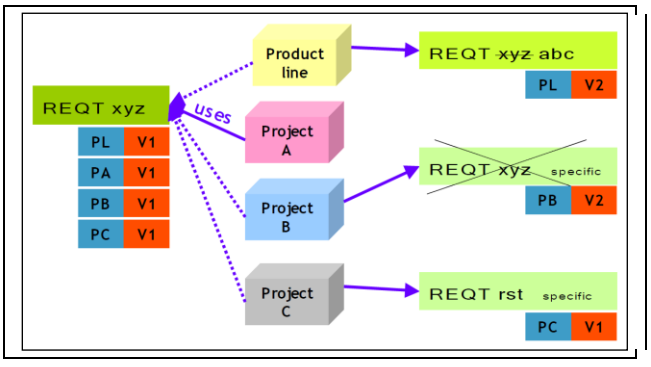

Figure 1: Three ways of reusing requirements

The NextGenRE Approach

From document centric development to model centric development. The System and Software Functional Requirements Techniques (SSFRT) study was sponsored by ESA in

2008-2009 to investigate the application of model based engineering technologies to support the space system development processes, from mission level requirements to software implementation through model refinements and translations. The primary result of the study was the Model-Based methodology to support Space System Engineering (MBSSE) and system / software co-engineering using SysML. As a true system engineering methodology, MBSSE deals with system requirements as first class citizens; they are captured, visualized and traced all along the modelling process. The MBSSE methodology was successfully applied to a case study in the context of the ExoMars mission and evaluated for integration with the ESA Concurrent Design Facility. It also provided the methodological point of departure for the model-based approach to requirements engineering pursued by NextGenRE.

Modelling and Ontologies. The issues presented in the section on current requirements engineering practice require a break from traditional approaches. Most systems are not developed from scratch, starting with a blank page. They are developed based on long experience and a history of developing previous systems. From the highest mission level requirements, a great number of low level requirements can be derived. In practice, the high level mission requirements also already fix a very large part of the design. If a satellite is part of the mission, then we already know a lot about the design of the satellite and its main subsystems. Furthermore, requirements and design elements at each level of decomposition are *tightly coupled*. This coupled decomposition of requirements and design elements goes down to the smallest screw (even if in most cases it is not carried through to the full extent).
The NextGenRE approach to improving requirements engineering involves supporting the reuse of *a priori* knowledge of requirements and design, best practices and lessons learned. This is done by using (semi) formal methods to describe systems and their properties, mainly using models and ontologies. This has two main consequences: First of all, it turns requirements engineering into a *knowledge management* problem. Secondly, it couples requirements and design. We have to look not separately at requirements or products, but at knowledge of the system as a whole and over all phases. In other words, a single (logical) data repository must contain *all* available information about a system over the whole life cycle. The data repository has to enable the establishment of relationships between different elements of all phases, as well as their manual and automatic processing. To make this even more challenging, the knowledge management system has to handle information in natural language (large parts of the requirements and documentation) and also information in formal models, expressed in languages like SysML.

The NextGenRE technical approach. To meet the challenges of this integrated knowledge management system we have chosen a combination of semantic wikis and modelling. A wiki makes it possible to organise textual information in easily editable pages linked by hyperlinks. This textual information can be entered as free text or by using forms and templates. The pages are identified by their URLs, and can therefore be seen as resources in the sense of an ontology. In other words, they can be used as subjects and objects in the triple statements of the Resource Description Framework (RDF) / Web Ontology Language (OWL). It is therefore possible to make statements about pages, for example to relate them to other pages or to define properties for them. If, for example, we have the pages:

- http://www.example.org/ExoMars/SpacecraftComposite
- http://www.example.org/ExoMars/CarrierModule
- http://www.example.org/ExoMars/DescentModuleComposite

then using a simple wiki, we can now define hyperlinks from the SpacecraftComposite page to the CarrierModule and the DescentModuleComposite, to say in human language that the SpacecraftComposite consists of the CarrierModule and the DescentModuleComposite. A user looking at the page SpacecraftComposite can use these hyperlinks to navigate to the two

pages, to obtain more information about them. But a computer processing these pages would not be able to understand (without very complex processing of natural language with all the associated issues) how these pages are related to each other, nor how the elements they describe are related to each other.
This problem can be solved by *semantically linking* the pages, using statements expressed in RDF. A statement always has the form <Subject> <Predicate> <Object>. The Subject is the page on which we define the relationship. The <Object> is the target of the relationship. Both are expressed as URIs, so they can directly reuse the URIs of wiki/web pages. <Predicate> describes the type of relationship, for example "consistsOf". Now we can add the following statements to the page:

http://www.example.org/ExoMars/SpacecraftComposite
http://www.example.org/ExoMars/SpacecraftComposite consistsOf
http://www.example.org/ExoMars/CarrierModule
http://www.example.org/ExoMars/SpacecraftComposite consistsOf
http://www.example.org/ExoMars/DescentModuleComposite

Now a program processing these pages can understand the relationship between the pages and the elements they represent. In addition, it is also possible to add properties to the pages, e.g. to express that the CarrierModule has a specific mass or power consumption. In a similar way it is now possible to define relationships between requirements, e.g. "derivedFrom", or between a requirement and a design element, e.g. "isSatisfiedBy".

Semantic technologies versus modelling. At first glance, the question may arise why we are integrating wiki, semantic technologies and current UML/SysML-like modelling. Semantic technologies and UML/SysML in particular seem to be very similar in what they provide. To a certain degree this is true, but there are important differences. Semantic technology is firmly rooted in formal logic, where UML and SysML are founded on practical object oriented software engineering. As a consequence automated reasoning is possible for properly constructed semantic datasets, whereas this is not the case for typical UML/SysML models. Currently, there is a very strong interest in the *integration* of both approaches in order to leverage the strengths of both. For example, there are recent efforts to transform between Eclipse EMF and RDF/OWL or to use a triple-store for storing EMF models (Google 2011).

Superior text management. While both UML/SysML-like models and ontologies provide very important functionality, some key questions remain: what to do with longer texts? How to edit textual information? How to display it in a user friendly form? This is a weak point in most modelling tools. For example, the handling of text in SySML requirements is not yet well-developed. Ontology development and browsing tools are not much better; they usually handle text as literals. Here, the wiki technology comes into play: it is a very convenient and simple way to edit and display even large amounts of text. A wiki page is directly addressable as a URI, which can be seen as a resource descriptor of the ontology world, or which also could easily be added to a modelling element. Using a semantic wiki, it is also possible to integrate the wiki into the semantic world, and to reuse a large number of related tools and libraries. Furthermore, for the end-users it provides an attractive way to bridge between the traditional document-centric views and the new model-centric views. This allows us to use a semantic wiki as a user interface for the mainly textual information of a knowledge management system.
Choice of semantic wiki technology. Semantic wikis are not new; there are already a number of existing implementations. But they all turned out to be not very well suited for our needs, especially when taking into account a long-term roadmap towards an integrated, collaborative knowledge management tool meeting the very demanding requirements of

distributed and integrated mission and spacecraft development. XWIKI, a second generation open source wiki implemented using a Java application server, was chosen for NextGenRE. In addition to the standard text editing capabilities of a wiki, it provides a number of features most helpful for using it as a general purpose, adaptive semantic collaboration platform. These features enable the implementation of important functionalities in the NextGenRE requirements engineering tool. In particular, the combination of GUI integration and Remote Procedure Call (RPC) access is extremely powerful. It becomes possible to fully integrate the wiki user interface with the modelling GUI: the user clicks on a modelling element in the modelling GUI, and then obtains the related documentation displayed in the wiki window, neatly formatted, with all related discussions, links to additional information and so on. On the other side, it is also possible to generate wiki pages from models or ontologies, for example to produce a skeleton for well structured documentation of classes and their attributes or to present the output of model to text transformations (e.g. for accreditation and V&V) in a user friendly way. In addition, using the wiki's scripting capabilities, it also would be possible to integrate additional tools, e.g. OpenModelica and ModelicaML. This would make it possible, for example, to analyse physical properties of systems, like budgets.

XWIKI Semantic Extensions. Figure 2 shows the overall architecture of XWIKI with semantic extensions, and both web browser and XEclipse as user interfaces. It consists of the XWIKI core running on a Tomcat application server, the semantic extensions, and the semantic storage, SDB and PostgreSQL.

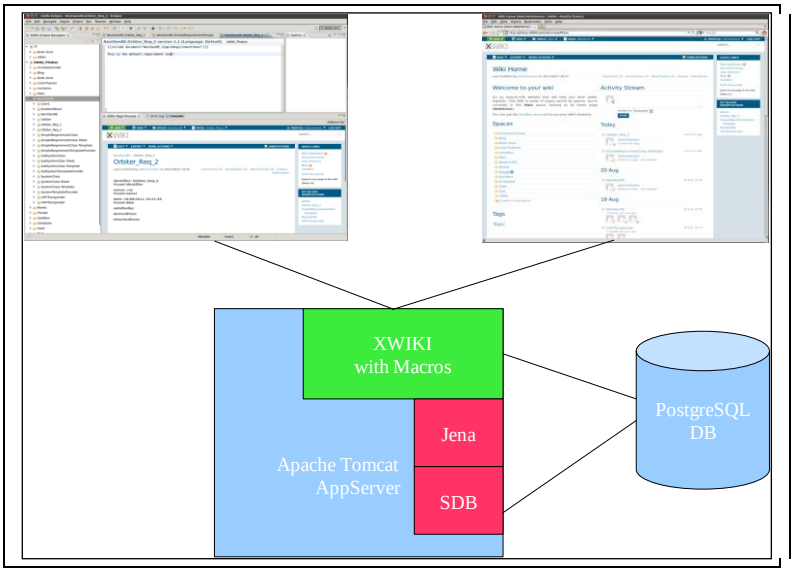

Figure 2: XWIKI with semantic extension and XEclipse Architecture

As a first step towards XWIKI-based requirements engineering and management, we implemented two main generic components: *Semantic macro* support and *semantic templates* using XEclipse. At first glance the *semantic macro support* (SMS) for XWIKI is very similar to the facilities provided by other semantic wikis, such as SemanticMediaWiki, since it also makes it possible to add semantic information to wiki pages. For example,

```
{{semProp mode="ADD" value="http://www.esa.org/subsystems::http://www.esa.org/NextGenRE/UHFTransponder"/}}
```

on the ExoMarsRover pages defines that the ExoMarsRover has a subsystem UHFTransponder. But it is implemented in a completely different (and much more powerful) manner. SMS is based on a mainstream ontology library, the Jena Ontology API. This library is reused in the form of macros, which are embedded in the wiki pages. SMS therefore

inherits the full functionality of the Jena library. This includes simple generation of resources and properties as in SemanticMediaWiki. But, in contrast to SemanticMediaWiki, it also enables advanced capabilities like SPARQL queries, full ontology support, reasoners, and import/export in formats like RDF or OWL. For example, an embedded SPARQL query is:

```
{{semQuery query="select ?id where {
?url <http://www.esa.org/NextGenRE/identifier::is_simplerequirement>  'TRUE' .
?url <http://www.esa.org/NextGenRE/identifier> ?id .
?url <http://www.esa.org/NextGenRE/status>  'approved'
}" header="id" links="true" linksAttrs="ref>id"/}}
```

This query lists all already-approved requirements. Using a standard web browser, this is then displayed as shown in Figure 3.

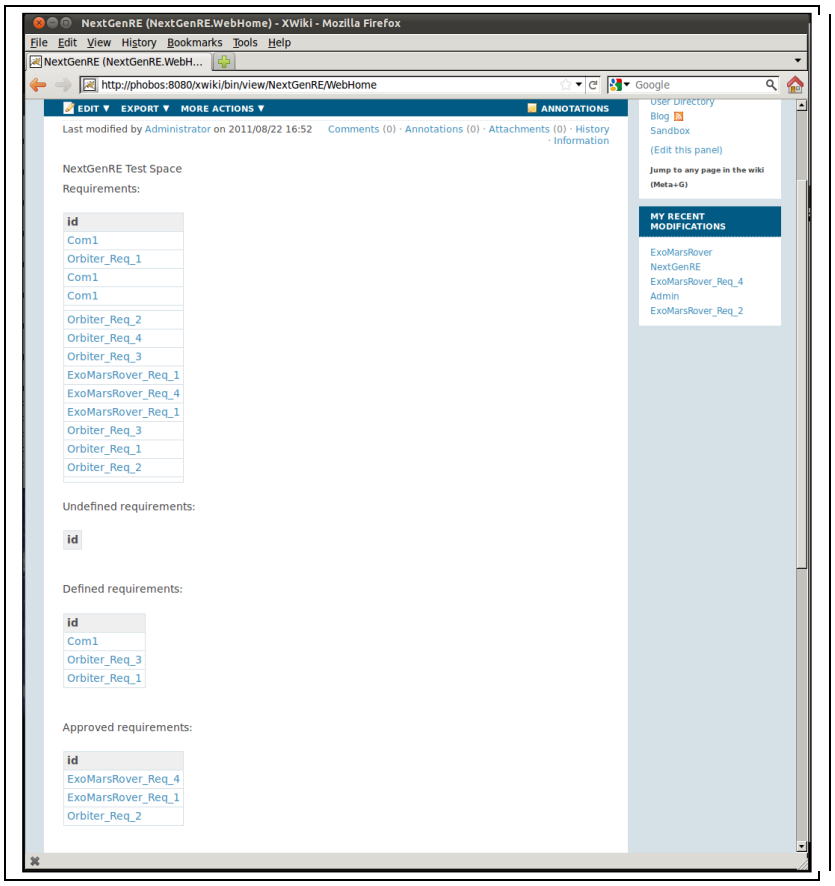

Figure 3: Results of embedded SPARQL queries

From the results of the SPARQL query, it is possible to directly navigate to the pages representing the information, just by clicking on the entry.

Jena strictly conforms to the main standards of semantic technologies. Therefore, it can be used in the context of semantic technologies in general. In fact, multiple software packages could be used to feed the semantically augmented XWIKI, because they all support the relevant standards and are fully compatible.

The semantic support in XWIKI can be tightly coupled to the XWIKI classes and objects. Objects (instances of classes) can be tagged to pages. They contain structured information, e.g. the owner and status of requirement pages. It is easily possible to attach these objects to semantic information. Whenever a property of a page is modified (e.g. the status of a

requirement is changed) this is directly reflected in the semantic information.

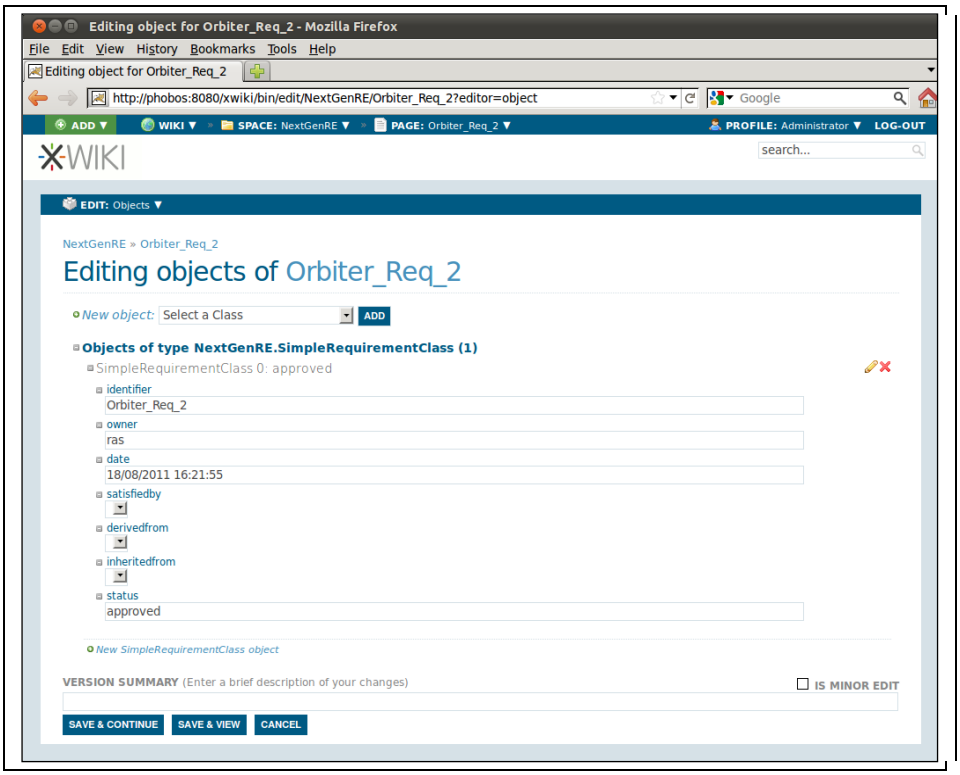

Figure 4: Editing an object tagged to a requirements page

Editing semantic information as tagged objects is very useful, for two reasons: First of all, the structure of the information can be well defined in a class associated with a template. Secondly, it is much easier to just enter a string or number (or to select from choices) than to enter a full semantic macro.

As persistent storage for semantic information, we use SDB with PostgreSQL 9 as a back end. The semantic storage can also be accessed by other tools, not only by the XWIKI macros, e.g. to add, modify or remove triples, but also for inference or any other processing. PostgreSQL is also used as back end storage for XWIKI. Therefore, the user has to maintain only one RDBMS, e.g. to backup both semantic and page related content in one operation or to migrate from one host to another.

XWIKI Semantic Template Support. XWIKI with SMS macros constitutes the SemanticXWIKI (SemXWIKI). SemXWIKI is already a very powerful generic knowledge management system. But it is hard to define pages from scratch all the time, especially if the pages are highly structured and need to have specific properties. This is especially the case for requirements. Therefore, we implemented a facility for the generation of XWIKI pages with related semantic content based on predefined templates. Instead of starting from an empty page, the user is provided with a form to enter predefined information. This information, which includes properties in XWIKI objects and semantic information, is then used to generate the page and the related content of the semantic database.

Eclipse Integration. XEclipse is an XWIKI GUI directly integrated with the mainstream Eclipse GUI. It is used as an alternative user interface for XWIKI and has several advantages compared to standard web browsers. First of all, it provides an integrated user experience. The system developer has only to interact with a single Eclipse instance, instead of multiple programs. The main advantage is the simple interface for page editing, especially pages with semantic queries. The XEclipse GUI has two main windows (the source window and page

preview window) and an explorer for space and page navigation on the left (Figure 5).

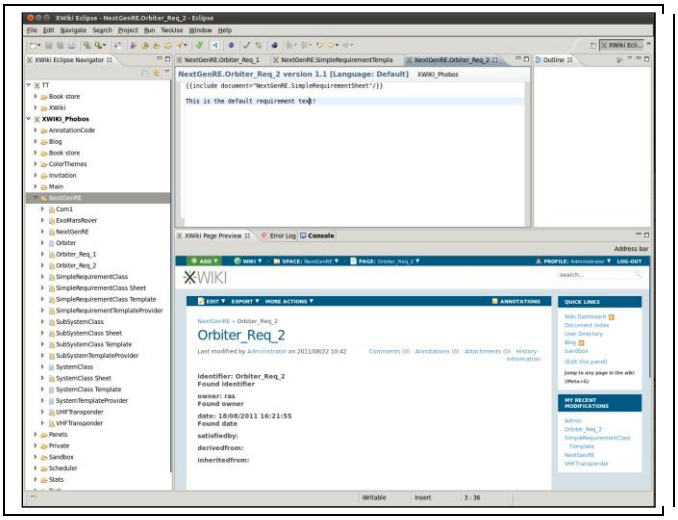

Figure 5: XEclipse GUI

Whenever the user modifies and saves the source, the preview window is automatically updated, and all semantic operations are executed. In particular, this includes SPARQL queries. So the user sees both query and results at the same time. In our practical experience, this is a very user friendly way to work with SPARQL queries, which then for example could be directly copied into Java programs. On the left side of Eclipse, the XEclipse Browser shows all spaces and pages. It allows direct navigation to pages, but navigation inside the preview window is also possible. XEclipse supports all editing capabilities of XWIKI, including classes and objects, in the same way as in a browser.

Template-based requirement creation. In NextGenRE, XEclipse's main application is the generation of semantically augmented pages (e.g. representing requirements) from templates. Figure 6 shows the New Page Wizard, which is being used to generate a new requirement "Orbiter_Req_4" from the template "NextGenRE.SimpleRequirementTemplateProvider". The target space is automatically set; the user has to add the name and title and to select the template to use. If no template is selected, a simple empty XWIKI page is generated.

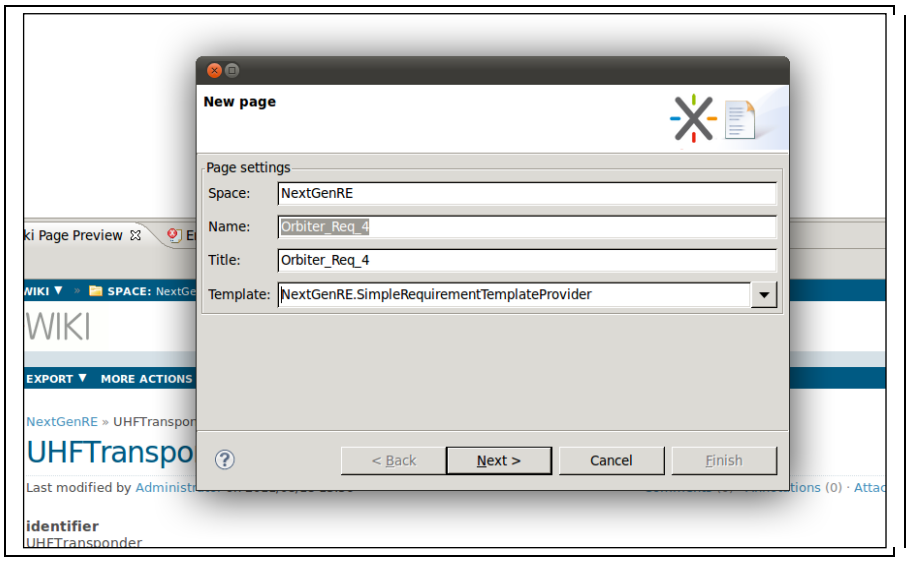

Figure 6: Starting the New Page Wizard

Figure 7 shows the new requirements page in source and in a preview, generated by the XEclipse plugin. It contains a text template, which in a more complex template might also include semantic macros, and the object properties of the class defined in the template. But object properties and semantic information from additional macros are automatically stored in the semantic database.

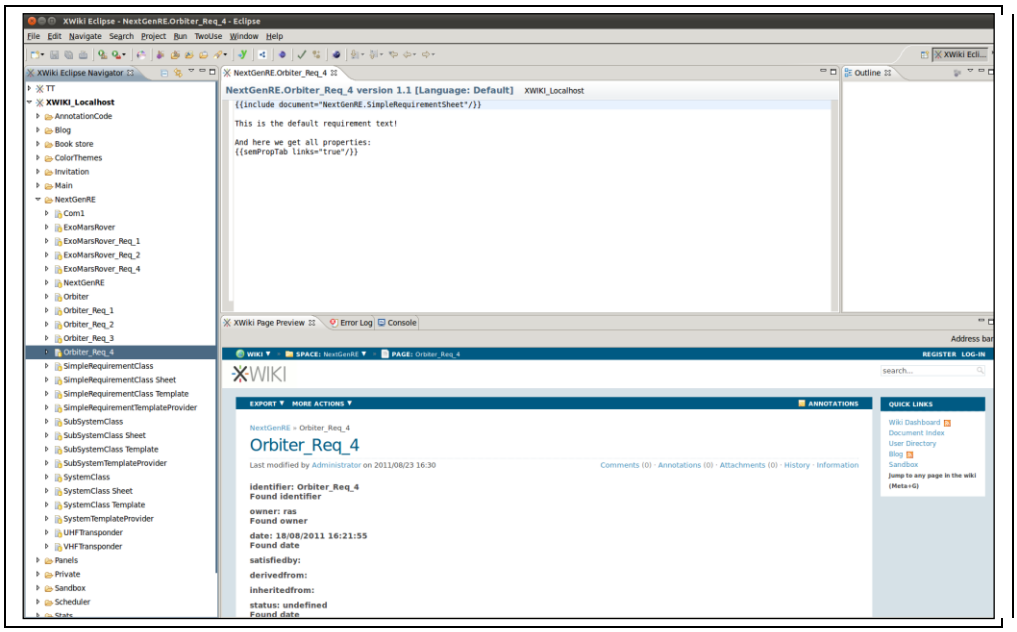

Figure 7: The new requirements page

Requirements Interchange Format (ReqIF). For next generation requirements engineering, it is necessary to provide a bridge for migrating the enormous repositories of requirements already in the possession of prospective users, but managed with existing proprietary or commercial tools. We have implemented an import / export facility supporting the Requirements Interchange Format originally proposed by the Germany automotive manufacturers' association and later taken up by the Object Management Group (OMG April 2011). This interchange format is supported by commercial tools – in particular recent versions of DOORS, the most frequently used tool in the Space industry. For this implementation we used modelling technology provided in the Eclipse Requirements Modeling Framework (Eclipse Foundation 2011), which is tracking the evolution of the Requirements Interchange Format.

SysML Gateway. A gateway to a SysML environment has been implemented in order to enable a strong coupling between requirements and design. It is possible to drag the requirements from the XWIKI Eclipse Navigator to a SysML editor (e.g. Papyrus). This creates a SysML requirement both in the graphical editor and in the SysML model underneath. At this point it is possible to use it in the model with the tools of the graphical editor, for example to trace a *Satisfy* association to another component, as shown in Figure 8. Future extensions will include the storage of canonical architectures (e.g. overall space system designs) in the repository in order to elaborate the capability for reuse of requirements in parallel with the reuse of designs.

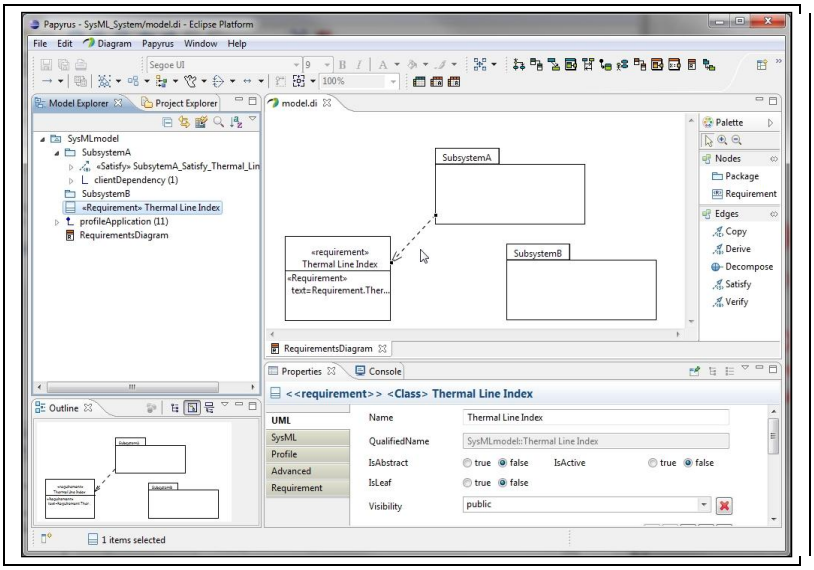

Figure 8: Coupling of requirements to SysML design elements

User Feedback and Discussion

Early versions of the tool have been deployed in the system engineering user community within NextGenRE partner Thales Alenia Aerospace; preliminary feedback and discussions of experience acquired with the tool and approach are provided in this section.

Enhanced requirements engineering flexibility. Most of the time, projects do not start writing requirements directly in tools such as DOORS. A first iteration (during which the requirements are often modified) is performed with the support of office tools; only afterwards is the preliminary baseline is imported into a dedicated tool such as DOORS. In contrast, the NextGenRe approach makes it possible to start the requirement engineering process directly in the tools used throughout the project. The principal differences with DOORS concern the simplicity of tool management. XWIKI has features that permit the customisation of the requirement template and to define views on the requirements database. This customization can be performed by the managing system engineer, with is no need for dedicated skills; projects can therefore be autonomous in terms of requirement management.
The use of Web technology greatly enhances the ergonomics of requirements creation and editing. Traditional tools mainly allow only plain text entry or, in some cases, selection of values from predefined lists, and perform only an *a posteriori* syntax check to verify the contents of a field. In contrast, the Wiki and Web technologies of NextGenRE support numerous enhanced features for requirements engineering, such as radio buttons, check boxes, and the possibility to change parts of the template dynamically according to values that are entered by the user.
In this way, a requirement description template is obtained which has a generic part and possibly specific parts according to the values of fields. For examples, the generic part might contain an ID, a textual description, a version, owner, and so on; specific parts might be defined for certain non-functional requirements (e.g. performance or safety related), which generally involve additional information compared to functional requirements.

Semantic navigation. Navigation between requirements using semantic information is of great interest. Traditional requirement engineering tools generally offer one-to-one level navigation, often keyed by requirement identifiers. In the NextGenRE environment it is more dynamic, and a second dimension is added through the semantics of the requirements.

Dual environments. The duality of a Web environment and an Eclipse environment adds

value to the requirement engineering process: on the one hand, the Web environment provides a simple way of entering textual requirements that are easily accessible to all system engineers. On the other hand, the Eclipse environment makes it possible to link the textual requirements with model artefacts, representing the requirements model as well as the architectural model. This duality provides a path for a smooth transition from today's mainly document-centric practices to tomorrow's model-oriented practices.

Legacy requirements. Every company involved in space system engineering already has a strong background of formalised requirements. The NextGenRE tool supports importing this material under the form of Office documents or DOORS databases using ReqIF.
The evaluation by Thales Alenia Space of the NextGenRE tool and methodology has resulted in the impression that this combination of Web and Wiki technology and a modelling approach to requirement engineering represents a promising new direction for requirement engineering that observes all the rules and constraints required for good product quality while at the same time improving the daily work of the system engineering by allowing him to focus only on technical matters rather than superfluous administrative details.

Related Work and Future Directions

Related work. Our approach is in line with efforts at NASA JPL to integrate ontologies and modelling (Rouquette and Jenkins 2010). JPL is developing a formal OWL ontology for flight project development to formally capture knowledge, e.g. to normalise terminology and improve data exchange between systems. These concepts from the ontology domain have to be transferred to the MBSE/SysML domain, e.g. to express precise semantics in SysML. The JPL work includes layering of ontologies (foundation, discipline, application) and a clear structure of systems (components, interfaces, function, ...), which is a sound base for our approach.
A close integration of our ongoing project and these efforts could be most beneficial for both sides. We could reuse the ontologies and tools developed by JPL. On the other side, the semantic wiki could be reused in the JPL project. For example, it is not difficult to generate well-formatted, readable wiki documentation directly from OWL models, which then supports all wiki features such as discussions or document generation.
On the requirement side, the JPL project is reusing the requirements concept of SysML, which is weak. It does allow a structuring of the relationships between requirements, but the content of a requirement is still a simple string. Here, the JPL project could be complemented by NextGenRE. The JPL work delivers the "vocabulary" for the NextGenRE approach, in the form of a domain ontology, while NextGenRE provides an integration of this vocabulary with templates, the association of requirements/requirement templates with design elements and the transformation from requirement templates to requirements. This will expand the "semantically accessible" domain of knowledge to the requirements, and will enable traceability and automatic processing from requirements to design.
A concrete and easily accessible description of using knowledge management and ontology technologies in spacecraft design is (Tahera 2009). Here, a methodology-based knowledge management was used for the design of an effective Attitude Determination and Controls (ADCS) subsystem.
In the following, two areas of future directions of investigation are briefly indicated:

Natural language processing. In practice, the biggest barrier to the uptake of semantically enriched requirements is the entry of the requirements themselves. The existence of open requirements interchange formats does not solve the problem, because the real problem is the semantic tagging of the natural language requirements. Natural language processing has been considered in requirements engineering in the past (Bucciarone et al. 2005), but it has not

been focused on providing efficient techniques for semantic tagging, possibly automated, on a large scale. Fortunately, this same problem is being addressed on a massive scale in other domains today because of recent advances in machine learning. Entities all over the world from newspapers to corporations are faced with the problem of adding semantic properties to their vast archives of text, in order to enable semantic search and other types of data mining. Automatic or semi-automatic semantic tagging technology is beginning to emerge in the framework of natural language processing, and a future extension of NextGenRE will take advantage of this technology to enable not only efficient import of requirements (e.g. through the ReqIF) but also their efficient semantic tagging.

Parametric requirements. Requirements analysis activities, particular of quantifiable capability and performance requirements, mainly address the transformation of the initial textual requirements into verifiable formal constraints. One of the most burning issues is the ability to refine textual requirements into parametric constraints expressed as mathematical equalities or inequalities that can be directly mapped to design elements and then to compute their satisfaction in a modelling environment. Here the challenges are somewhat simpler than in requirements expressed in natural language, because mathematical expressions are much more formal than natural language. The main challenge is the well defined relationship of the variables of the expressions in the requirements to model elements of later phases. It may be solved by a metamodel for mathematical expressions which is able to express the semantics of the expression. Using structural editing or a compiler, this metamodel can be populated directly from mathematical expressions in a machine readable format. At later phases, the elements of this metamodel expressing the "entities", the variable of the expression, would be mapped to corresponding elements of product models. Work to achieve this is underway with the (ECSS-E-TM-10-25) technical memorandum. This will ultimately enable computer aided verification of the requirements satisfaction.

Conclusions

The transition of requirements engineering and management (REM) from the practices of today to those of the next generation requires new approaches along multiple dimensions. The move from a document-centric focus to a model-centric focus provides support for the automation of key activities all along the REM lifecycle. The introduction of dual semantic wiki and SysML modelling environments enriches the user experience and enhances the expressive possibilities of the requirement creation process, while providing an explicit acknowledgment of the tight coupling between requirements and design in practice and a means to deal effectively with it. The adoption of an open requirements interchange format provides an essential bridge in the transition from the old to the new.

References

Bucchiarone, A., Gnesi, S., and Pierini, P. 2005. "Quality Analysis of Natural Language Requirements: An Industrial Case Study". Paper presented at the 13th IEEE International Requirements Engineering Conference (RE05), Paris, France, 29 August – 2 September.

Decker, B., Ras, E., Rech, J., Jaubert, P., Rieth, M. 2007. "Wiki-Based Stakeholder Participation in Requirements Engineering," *IEEE Software*, vol. 24, no. 2: 28-35.

Eclipse Foundation. August 2011. "Requirements Modeling Framework Project." Accessed 02 November 2011. http://www.eclipse.org/rmf/

Estefan, J. 2008. "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Revision B. http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf

Google. 2011. “eclipse Labs emftriple: (Meta)models on the Web of Data.” Accessed 02 November 2011. http://code.google.com/a/eclipselabs.org/p/emftriple/.
Hull, E., Jackson, K., and Dick, J. 2005. *Requirements Engineering* (Second Edition). London: Springer Science+Business Media.
Object Modeling Group. February 2011. Systems Modeling Language (OMG SysML™), SysML v1.2 Specification.
———. April 2011. Requirements Interchange Format (ReqIF), ReqIF v1.0.1 Specification.
Robertson, S., Robertson, J. 2006. *Mastering the Requirements Process* (Second Edition). London: Dorset House.
Rouquette, N. and Jenkins, S. 2010. “OWL Ontologies and SysML Profiles: Knowledge Representation and Modeling.” Paper presented at the OMG Eclipse Symposium.
Tahera, K. 2009. “Development of Knowledge Based Tool to Assist Spacecraft Control Subsystem Design.” Master’s Thesis at Lulea University of Technology.

Biography

Hans Peter de Koning is a system engineer in the European Space Agency’s Systems, Software and Technology Department. His main responsibilities are the methods, tools and standards for the Concurrent Design Facility at ESA/ESTEC. He received his M.Sc. degree in physics from Delft University of Technology, and has more than 25 years of engineering experience in industry and at ESA. He is an active contributor to INCOSE’s MBSE Initiative.

Silvia Mazzini is Methodologies and R&D Manager at Intecs SpA. She is the coordinator of the CHESS ARTEMIS-2008-1-100022 project, addressing composition with guarantees of multi-concern components for high-integrity systems. She also was leader of several studies for ESA/ESTEC and participated in the SESAR Consortium. She took her degree in computer science at the University of Pisa.

John Favaro is a senior consultant in the consulting division of Intecs SpA, where he is also Deputy Director of Research. His current interests are oriented toward safety critical systems engineering. He has degrees in computer science from Yale University and the University of California at Berkeley.

Rudolf Schreiner is the CTO of ObjectSecurity Ltd. in Cambridge, UK. He studied physics and astronomy at University of Munich. His main interest is the Model Driven Development of complex, secure systems in mission critical domains, especially in aerospace and defense.

Xavier Olive is a research engineer at Thalès Alenia Space France in the research department and satellite and platform section. He received a Ph.D. degree in model based diagnosis (control and computer science) in 2003 from Paul Sabatier University in Toulouse, France and an Engineering Degree from Ecole des Mines d'Alès, Alès, France, in 2000. He is in charge of research activities about satellite autonomy and model based engineering at system and software levels.