Modelling Requirements in Service to PLM for Long Lived Products in the Nuclear Field

Albéric Cornière, Virginie Fortineau, Thomas Paviot, Samir Lamouri, Jean-Louis Goblet, Audrey Platon, Cécile Dutertre

Modelling requirements in service to PLM for long lived products in the nuclear field

Albéric Cornière*, Virginie Fortineau*, Thomas Paviot*,
Samir Lamouri*, Jean-Louis Goblet*, Audrey Platon**, Cécile Dutertre**

*Arts et Métiers Paristech, 151 bd de l'hôpital, Paris, France
**EDF - DIN - projet PLM, 97 av Pierre Brossolette, Montrouge, France

Abstract. Requirements engineering in usually considered a first step before design that is to evolve with each generation or version in a product line. Nuclear plants however, are subject to modifications during their lifetime, in their design and implementation as well as in the requirements they have to satisfy. Economic, technical and safety reasons lead to extending the requirements engineering process through the whole life-cycle of the nuclear plants. This article presents an ontology-based approach to integrating the requirements engineering into a PLM approach for such long-lived, large-scale products.

Keywords: Requirements Engineering, PLM, Ontology, Long Life-Cycle, Large-Scale, System Engineering

1 Introduction

Requirements engineering in usually considered a first step before design that is to evolve with each generation or version in a product line. Nuclear plants however, are subject to modifications during their lifetime, in their design and implementation as well as in the requirements they are to satisfy. Economic, technical and safety reasons lead to extending the requirements engineering process through the whole life-cycle of the nuclear plants. This article will present the requirements engineering with the goal of setting a frame to the problem, then it presents the different approaches that are used, mainly in the field of computer sciences. In section 4 several particularities in the context of nuclear plants are given, leading in section 5 to a proposition of modelling with the intent to answer the industrial problem.

2 Definitions and uses of requirements

Requirements engineering is commonly mentioned in scientific publications, especially in the field of computer sciences. However, there is seldom a definition given for this process, and comparatively few of the articles available treat of requirements engineering itself. Jureta [13] notices "To say that requirements are engineered is currently more of an ideal than the actual state of affairs". We

will retain the definition given by [6] : "Requirements engineering is the process producing a coherent set of specifications on a yet-to-be-designed object"

As for requirements, the definition is still heterogeneous depending on the context in which the notion thereof is used : in most situations, the starting point with requirements engineering is the expression of what the product is to accomplish, that is to say the expression of the needs of the stakeholders. The SysML specifications [15] refer to requirements in the fields of computer sciences as thus : "A requirement specifies a capability or condition that must (or should) be satisfied. A requirement may specify a function that a system must perform or a performance condition a system must achieve", while many other consider them as high level specifications of what the product is to accomplish [1]. We were unable to find in the scientific literature a situation presented in which requirements are considered outside the beginning of life of a product, or in which they may change during the product life-cycle.

For the purpose of this article we will consider a requirement as "one or several properties or behaviours of a system that must be satisfied"[15]

3 Requirements modelling today

Recent works using requirements engineering are mostly from the field of computer sciences ; about 90% to 95% of the search results on a scientific repository such as Springer link for "requirements engineering" are identified as computer science works. In those works, two main tendencies can be identified : on one hand, works that focus on defining the requirements so as to waive ambiguities, in order to allow design to be based on a reliable transcription of the stakeholders needs; and on the other hand works that focus on analysis of the requirements sets, in order to ensure the requirements set is coherent and satisfiable.

3.1 Eliciting the requirements

The first goal of requirements engineering is to define what needs the stakeholders express : several studies focus on this task [8, 17, 11]. Goal-driven requirements engineering addresses this question by defining goals to be attained during design, and prioritising them considering which stakeholder expressed it, what importance it has to the main goals, and several other criteria.

The elicitation problem is crucial : even in the nuclear context and considering safety rules and regulations that are not to be interpreted, the requirement still have to be translated into expressions in a model for a computer to manipulate them. The elicitation of requirements is therefore needed to ensure a proper correspondence between expressed requirements and their expression in the model. [13] presents a possible use of the DOLCE ontology (from [14]) for requirements elicitation.

3.2 Analysing the requirements

As most, if not all requirements define an obligation (or an interdiction) for the system, it has been shown by [2, 6, 7, 12, 13] that a modal logic can be defined that allows to treat requirements as logic expressions expressed in first-order deontic logic. From there on, the whole set of requirements can be analysed through formal logic, manipulated and corrected to render the requirements set consistent if needed.

3.3 Modelling requirements with ontologies

Ontologies, especially using OWL2, allow for reasoning in direct logic: an ontology designed for requirements analysis is presented in [13]. This model based on goal-driven requirements engineering is thorough and reliable, although it is thought primarily for the design phase of a product's life-cycle.

In the life-cycle of a product family, this presents little problem, as the evolution in requirements are addressed through different versions of the product; the requirements engineering process may then be repeated for each version of the product.

For the context of the nuclear industry, [5] shows the benefits of ontologies in modelling products and uses rule-based models to express business rules, which in the model are similar to requirements.

4 Specificities of a PLM context in the nuclear industry

Modelling requirements for a nuclear plant in a PLM context presents specificities, among which some regarding the scale and complexity of the system, the length of the life-cycle, and the type of requirements considered.

The life-cycle of a nuclear plant, from the beginning of the requirements engineering phase to the end of dismantlement may well exceed a century. During this time-span, advances in the technology as well as experience in the field makes the requirements evolve, as well as the technical solutions available to satisfy them. While works exist regarding the evolution of requirements, they mostly focus on requirements that apply to software systems[4].

A nuclear plant is of large scale : the specifications elaborated through the requirements engineering process apply on billions of parts and systems. All those elements of the product are potentially interacting with each other in non-trivial ways. This makes the system too complex for manually going through the requirement engineering again. One of the characteristic examples is thermodynamic evolution of the reactor : the temperature and pressure inside the reactor can vary slowly in comparison to the speed at which a computer program is usually running.

The requirements on a nuclear plant include safety regulations that specify situations in which the plant has to return to a controlled, stable state when an accidental situation is to arise. This kind of requirements must be satisfied in any configuration, regardless of the evolutions the requirements and of

the maintenance on the plant, and through the evolutions of the plant itself. These requirements typically specify the time acceptable to return in a controlled state. As described above, the behaviour of the plant is not trivial, and complex behaviour and business rules may apply to the requirement to lead to the specifications.

To respond to the above-mentioned points, there is a need for a model allowing not only to define a coherent set of requirements, but also reference the specifications inside an as-complete-as-possible model of the plant, including what elements of said plant the specifications apply to. This model also must allow the engineers to make requirements evolve. This evolution can be in the form of new requirements being added over time, as well as changes to existing requirements.

5 Proposal of modelling approach

5.1 Generic modelling of requirements

Models based on ontologies have the expected benefits of completeness of information, due to the language expressivity, and of embedded intelligence thanks to the reasoning abilities associated with it [5]. There is also a need to model links between a requirement itself and several other elements of the requirements engineering process, namely the logical statements composing it, both premises and conclusions; the reference phrasing of the requirement, be it a legal document or a rule known by experience; the deontic nature of the requirement, whether an obligation or an interdiction[1];the specifications they contribute to and information on their validity. These links can be modelled in the form of object properties, to allow for a diversity of relations between the elements, rather than using classes.

To allow for this, it is necessary that requirements, logical statements, deontic functions, business rules, specifications, etc. are modeled as individuals in several classes[2]. Those classes are not necessarily all disjoint (for example a business rule can also be a requirement), thus making use of the non-canonic representation of data in ontologies.

An example of such a model is given in Fig.1 : A requirement is modeled as an individual. Its nature is defined by the link it has (`is_a`) with a member of `SDL_function`, in this case the interdiction function of FOSDL[6]. Its predicate are also identified with object properties (`has_premise` and `has_conclusion` in this example) and are FOSDL object expressed in RIF[16]. It should contain a data property representing the origin and a verbatim of the reference document it is translated from, for verification.

[1] According to [12] and [7] a requirement might also be any function built from obligation (O), interdiction (F), necessity ($\Box$), possibility ($\Diamond$) and negation ($\neg$)

[2] The reader should remain aware a class in ontologies is representative of a concept, an open set of individuals, rather than a generator thereof[3]

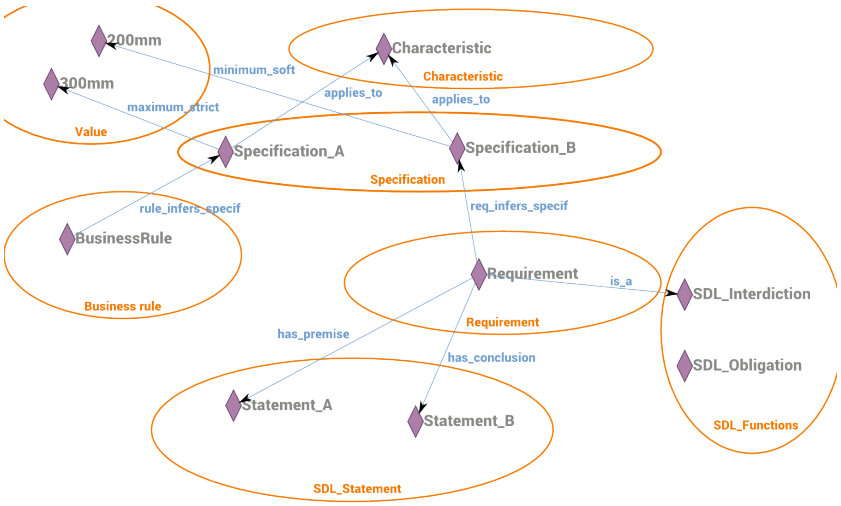

Fig. 1. Schematic representation of the object properties network around a requirement individual : individuals are represented by diamond-shaped symbols, classes by ellipse-shaped regions in the figure

The specifications are also represented as individuals, in their simplest form they link a `Value` individual to a `Characteristic` individual thanks to object properties. The choice to model the values themselves as individuals is intended to allow reasoning through the object properties themselves rather than to rely on the data properties, and to open a way for the logic refining of the individuals network, so that individuals can be treated as identical[3].[10]

The links from a `Specification` to a `Value` can, propagate the constraint to the `characteristic` because of the inferences the ontology-based model is capable of. Examples given in Fig.1 are inspired by goal driven requirements engineering and represent soft or strict constraints corresponding to soft or functional goals. The `Requirement` is also mapped to the specifications it infers (through the design phase), in the same way a business rule is mapped to another specification (and may also pair with a requirement, or be composed of statements in the same fashion, links that are not represented on the Fig.1 for clarity). Lastly, the characteristic can be mapped itself to a measured value, completing the path from the formal requirements model, and the physical implementation of the system.

[3] that is to say, they share the same identity, as `200mm` and `20cm`

5.2 Case of a safety requirement : physical separation of electrical networks

For the safety of the plant, it is needed that the control system remain functional is case of an incident. This is achievable through a wide array of means, ranging from the protection of the plant's organs to the liability of elements under circumstantial stress. In this part we will present a possible way to model part of the requirement network associated with the physical separation of redundant electrical networks. Let us consider the requirement : "In case of an incident, the plant remains functional".

As stated above, we will represent this requirement as an individual `Req_1` in the *`Requirement`* class. The premise for this requirement is "in case of an incident" : this premise is modelled as another individual `Statement_1` in the *`SDL_statement`* class. `Statement_1` contains in its data properties the mention of the source sentence: "In case of an incident" and its translation as a RIF expression, namely `occurs(inc,ctxt),Incident(inc),Context(ctxt)` with *`Incident`* a class to which all incidents belong, and *`Context`* the class of contexts for simulation or domain-restricted requirements.

A link (in the form of an object property) is established from `Req_1` to `Statement_1` so that `has_premise(Req_1,Statement_1)`.

In the same fashion, a `Statement_2` individual is created in *`SDL_statement`*. It represents "the plant remains functional" in the above-mentioned requirement, with a RIF rule in the form `Context(c),Functional(sys),inContext(sys,c)`. This rule needs *`Context`* to be a class of contexts [4]; *`Functional`* is a unitary function (a class defined by axioms) qualifying the functional state of the system; finally `inContext` is an object property for evaluation or simulation set-up, describing whether a context applies to a physical individual. An object property is set so that `has_conclusion(Req_1,Statement_2)`. As it is mandatory, `Req_1` is also connected with an object property `is_a` to `SDL_Obligation`, in class *`SDL_Function`*.

Next step is to describe business rules derived from this requirement. For instance consider the business rule "Redundant electric systems are to be separated physically". It is a business rule for it is a consequence of a requirement in a certain domain: it applies only to redundant electrical networks.

This business rule will be modelled as an individual `BusRule_1`, itself connected with object properties to *`SDL_statement`* individuals, that represent it premise (`Electric(A),Electric(B),redundantWith(A,B)`, translating "Redundant electric systems") and conclusion (`physically_separated(A,B)`, translating "be separated physically"). `BusRule_1` is also linked with `SDL_Obligation` by a `is_a` property.

As the network is added to the model, more individuals are created : in this case exploring the `physically_separated` object property make apparent the need to define criteria to set or not this property between two objects.

[4] In the scope of this work, a context is a subset of the worlds defined by some known circumstances [3, 7]

This approach being individual-oriented, we chose to define the object property through a set of rules, one of them given below.

```
Location(place1),Location(place2),
    is_in(A,place1),is_in(B,place2),
    DifferentIndividuals(A,B),physically_separated(place1,place2)
    → physically_separated(A,B)
```

This rule only states that "distinct elements located in physically separated places are physically separated themselves". Such rules reflect the experience and knowledge of the modellers, they are the interface from the requirements engineering to the knowledge management. Another rule can reflect the notion that "elements sufficiently far apart are considered physically separated"[5], with a rule in the form :

```
Element(A),Element(B),DifferentIndividuals(A,B)
    distance(A,B)>FarApart
    → physically_separated(A,B)
```

At the time being however, there is no direct way we know of to quantify a binary function (an object property), that is associate a value to a relation between individuals. It is not possible to directly use the expression `distance(A,B)>FarApart`. This is prevented only by the current implementations of the model, that do not yet implement ternary functions; and as this can be worked around modelling the relation, it is not established yet if this solution is acceptable on large-scale models.

6 Conclusions and perspectives

Using SDL and RIF definition for rules and requirements allows to model them as individuals in an ontology, and to leverage the reasoning abilities associated with the existing models. The simple pegging of requirements to specifications is not sufficient to check for the consistency of a set or requirements, or to verify and infer which requirements constrain each specification, and for which reasons. Doing so requires to introduce into the model more information, reflecting the business experience.

While the modelling of requirements into ontological models still need to be studied, it is a reasonable assumption that the ontological model would need to interact with a knowledge model, as well as a business rule model and a model of the physical system, for consistency check, and to allow for a sufficient mapping between informations that are all used in system engineering. Future works will have to investigate the interfaces between the different models.

[5] Any reader familiar with the field of nuclear security may object this rule does not exist. We are well aware of this, and this made-up rule merely serves as an illustration of the limits that may occur because of an ontological modelling. Similar rules do however exist regarding minimal distance between electrical cables, for instance.

Although the global work-flow comes from the requirements and leads to the specifications, a complex system and/or a vaguely formulated requirements imply using knowledge that belongs primarily to the designer : for a model to check or validate on a design selection, the knowledge used to make this choice must be incorporated to this model, in a form that allows for the users to incorporate it themselves, thus effectively making an ontological model the explicit representation of a shared conceptualization[9].

References

1. Cheng, B.H., Atlee, J.M.: Research directions in requirements engineering. In: 2007 Future of Software Engineering. pp. 285–303. IEEE Computer Society (2007)
2. Cholvy, L.: Checking regulation consistency by using sol-resolution. In: Proceedings of the 7th international conference on Artificial intelligence and law. pp. 73–79. ACM (1999)
3. Corniere, A., Fortineau, V., Paviot, T., Lamouri, S.: A concept-based approach to modelling shared ontology-based models for industrial applications. In: Proceedings of the 19th IFAC World Congress (2014)
4. Ernst, N., Borgida, A., Jureta, I.J., Mylopoulos, J.: An overview of requirements evolution. In: Evolving Software Systems, pp. 3–32. Springer (2014)
5. Fortineau, V., Paviot, T., Lamouri, S.: Improving the interoperability of industrial information systems with description logic-based models - the state of the art. Computers in Industry 64, 363–375 (2013)
6. Garion, C.: Apports de la logique mathématique en ingénierie des exigences. Ph.D. thesis, Université de Toulouse (2002)
7. Garion, C., Roussel, S., Cholvy, L.: A modal logic for reasoning on consistency and completeness of regulations (2009)
8. Greenspan, S., Mylopoulos, J., Borgida, A.: On formal requirements modeling languages: Rml revisited. In: Proceedings of the 16th international conference on Software engineering. pp. 135–147. IEEE Computer Society Press (1994)
9. Gruber, T.: Towards principles for the design of ontologies used for knowledge sharing. International Journal of Human Computer Studies 43 (5), 907–928 (1995)
10. Guarino, N., Welty, C.: Handbook on Ontologies, chap. An overview of OntoClean, pp. 201–220 (2009)
11. Helming, J., Koegel, M., Schneider, F., Haeger, M., Kaminski, C., Bruegge, B., Berenbach, B.: Towards a unified requirements modeling language. In: Requirements Engineering Visualization (REV), 2010 Fifth International Workshop on. pp. 53–57. IEEE (2010)
12. Jureta, I.J., Mylopoulos, J., Faulkner, S.: Revisiting the core ontology and problem in requirements engineering. In: International Requirements Engineering, 2008. RE'08. 16th IEEE. pp. 71–80. IEEE (2008)
13. Jureta, I.J., Mylopoulos, J., Faulkner, S.: A core ontology for requirements. Applied Ontology 4(3), 169–244 (2009)
14. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Oltramari, R., Schneider, L., Istc-cnr, L.P., Horrocks, I.: Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology (2002)
15. OMG: Sysml v 1.3 (2012), `http://www.omg.org/spec/SysML/1.3`
16. W3C: Rif core dialect (2013), `http://www.w3.org/TR/rif-core/`
17. Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Transactions on Software Engineering and Methodology (TOSEM) 6(1), 1–30 (1997)