Formalizing standards and regulations variability in longlife projects. A challenge for Model-driven engineering

Nicolas Sannier, Benoit Baudry, Thuy Nguyen

Formalizing standards and regulations variability in longlife projects. A challenge for Model-driven engineering

Nicolas Sannier* **, Benoît Baudry** and Thuy Nguyen*
* EDF R&D – STEP, 6 Quai Watier BP49
72148 Chatou, France
{nicolas.sannier, nthuy}@edf.fr
** IRISA/INRIA, Campus Universitaire de Beaulieu,
35042, Rennes Cedex, France
{nicolas.sannier, benoit.baudry}@Inria.fr

ABSTRACT

Safety regulations and standards imposed by national regulators on nuclear power plant systems provide high-level requirements, recommendations and/or guidance expressed in natural language. In many cases, this leaves a large margin for interpretation, not all of which are acceptable to a given regulator. Currently the elements that lead to the establishment of acceptable/accepted practices are not always documented, nor are these practices formally modeled. When a new standard appears or when Electricité de France (EDF) has to discuss a standard with another regulator, there is no systematic process to build a practice.

Domain-specific modeling, traceability and variability modeling are Model-Driven Engineering (MDE) techniques that could address various aspects of practice formalization. This paper precisely defines the modeling issues that are currently faced by EDF when managing regulatory safety requirements, standards and practices. Then we review existing requirements modeling techniques to understand their benefits and limits according to EDF's needs.

Keywords: requirements, standards, regulations, safety, practice, variability, traceability, modeling

I. INTRODUCTION

Longlife critical installations such a nuclear power plants are designed and developed with safety concerns. The objective is to ensure an acceptably low level of residual risk regarding incidents or accidents, and that procedures do exist to face them. Finally, it also deals with minimizing the impact of such events on people, on the environment and on the installation.

To handle this safety concern, from design to licensing with respect to a given safety authority, a licensee must comply with the national regulatory requirements and guidelines, and often follows the requirements and recommendations of standards and/or technical codes to support their work.

Unfortunately, these documents only provide high-level guidance, leading both licensee and authority to seek a common interpretation of their contents. This common interpretation, which may evolve over time, defines the basis of the *nuclear safety practice* at a specific moment, in a specific country.

Unfortunately all of these relations between standards, regulations and practice are rarely formalized. This is an issue to understand the impact of the evolution of all these elements and also to keep a homogeneous and global trace of this knowledge, which is actually split and partially shared between several different human experts.

Model-based systems-engineering is a major jump from the natural language and document-centric approach to the formalized application of modeling to support the full system life-cycle, from requirements to decommissioning. Our research question is the following one: Can we use MDE to model the practice through the formal representation of the relations between standards, regulations and their interpretations?

In this paper, we aim to define the requirements management problem for the construction and modernization of nuclear power plants. We illustrate different facets of this problem through the historical analysis of the development of Instrumentation and Control (I&C) systems at EDF. Then we review some existing requirements modeling approaches and identify gaps that we will have to handle in our future works. The remainder of the paper is organized as following: Section 2 describes our industrial context and the lessons we can learn from three tightly linked stories regarding I&C systems developments. In section 3, we propose to define some of the variability dimensions, the relations we expect to find and our objectives regarding this problem of variability in practices and knowledge formalization. Section 4 proposes a quick roundtrip of the research domains that are crosscutting to our problem with a particular focus on how we can model requirements dependencies. We finally discuss around the paper and perspectives of our future works in section 5.

II. ON STANDARDS, REGULATIONS AND PRACTICES

A. EDF context

EDF (Electricité de France) is the national electricity provider in France and owns and operates a fleet of 58 nuclear power units. Besides the continuous maintenance and surveillance during operation, the systems of a unit may be replaced or upgraded during the periodic outages that are

necessary for refueling and inspection. Such changes are under close regulatory scrutiny and subject to approval from the concerned safety authorities.

Experience from the field, technological progress or societal evolutions may lead authorities to modify their expectations regarding the different systems. Consequently, they ask the licensee to justify the system's safety based on new or modified criteria.

Safety justifications are grounded on technical arguments justifying technological choices, design decisions but also rely on practices that have been accepted in previous projects.

B. Building practice, three short EDF stories

To better understand this context, we propose to tell three stories related to the Instrumentation and Control (I&C) systems of EDF nuclear power plants. We first briefly describe two concepts important to our problem background: the first is the concept of series; the second is the safety classification. After these stories, we finally highlight several considerations that appeared along them.

1) General background

In France, EDF nuclear power units are built in series. The units of a series have the same general design, with relatively minor differences to take account of the specific site constraints. For example, the cooling systems need to take consideration of whether the unit is on the seashore, along a big river, or along a small river. There are currently four main series: the "900MW" series has 34 units, the "1300MW" series has 20 units, the "N4" series (1450MW) has 4 units, and the EPR (Evolutionary Pressurized Reactor) series has one unit still under construction.

A large number of functions are necessary to operate a nuclear power unit, and guarantee its safety. Functions are categorized (A, B, C or Not Classified) based on their importance to safety, category A functions being the most important to safety, and Not Classified functions being those that are not important to safety. In parallel to functions categorization, the I&C systems are classified based on the categories of the functions they implement.

As mentioned earlier, each system important to safety (i.e., implementing at least one category A, B or C function) goes through a safety justification process. In general, the safety authority lets EDF propose and present its solution and then decides whether the solution is *acceptable* or needs to be improved.

2) Standards as parts of safety justification

a) Introducing innovation in I&C system developments

In the early 90's, EDF was busy designing and developing its new N4 series, including the corresponding I&C systems. Based on the experience of the 1300MW series, EDF decided to use digital, microprocessor-based I&C systems for nearly all I&C functions.

At that time, there was no written French regulatory requirements regarding digital I&C systems important to safety, and the only existing *international standard* to support this process was the International Electrotechnical Commission 60880 standard, released in 1986 (IEC60880-1986). This standard deals with software for computers in safety systems of nuclear power units. EDF's position was to apply this standard to develop the software of systems performing category "A" functions. In parallel, for the systems performing functions of a lower safety category (B or C), there was no guidance; and EDF had to propose a safety justification approach from scratch.

The French safety authority (ASN) and its technical support organization (IPSN) required that particular recommendations of IEC 60880 be mandatory and be considered as requirements. Also, after lengthy discussion, EDF and IPSN agreed on an *acceptable* practical interpretation for a number of 'fuzzy' requirements. This story highlights the role of standards, regulations and the way practice is built through the interpretation of the corpus of documents which are used for such projects.

b) Dealing with standards evolutions

Since 1986, a number of additional standards have been published. They address the general requirements for I&C systems (IEC 61513), safety categorization and classification (IEC 61226), hardware requirements (IEC 60987), and software for I&C systems performing category B or C functions (IEC 62138), data communication (IEC 61500), common-cause failure (IEC 62340), etc. During the same period, ASN/IPSN considered that with the N4 project, they had gained enough experience regarding software important to safety, and issued a high level regulatory document, "Basic Safety Rule: Software for electric systems important to safety" (RFS II.4.1.a - 2000). In this document, it is mentioned that, in some specific requirements, conformance to specific clauses of IEC60880 is considered an acceptable practice. However, the Basic Safety Rule has been issued before many of the other IEC standards were published, and therefore it makes no mention of them. It is now an unwritten understanding that these newer standards also constitute acceptable practices.

c) Managing evolutions from N4 to EPR I&C systems

The EPR program for the first unit of the series (to be built in Flamanville) followed the N4 program (advanced discussions started at the end of the 90s). The overall architecture and design for I&C systems is based on what had been proposed and accepted in the N4 program, with however a number of significant changes. Firstly, the EPR I&C is based on different products, many of those used for N4 being no longer available, and industrial alliances between suppliers having changed since N4. Secondly, a number of IEC standards issued since N4 (such as IEC 62138 for software of I&C systems performing category B or C functions) are now applicable.

However, the most significant change (as far as this paper is concerned) is the emergence of products and technologies that were not used for N4, and for which there is no precedent and no standard. In particular, this is the case for what is commonly called 'smart components' such

as smart sensors or smart power supplies. (They are said to be smart due to the presence of embedded microcontrollers that allow capabilities and functions that cannot be provided otherwise, but that raise the same concerns in terms of safety justification as software.)

For these emerging technologies, new safety justification approaches need to be proposed from scratch.

3) Standards interpretation and regulation in different contexts

Born from a European program, the EPR design had been expected to be built on several countries: France, Finland, the United-Kingdom, and the US in particular. The case of the UK is particularly interesting. The British safety authorities reference the same set of IEC standards as France. However, the acceptable practices on both sides of the Channel differ on some significant points and lead to differences in I&C architectures.

In particular, safety approaches in the UK rely in large part on probabilistic approaches, whereas in France probabilistic analyses are considered as complementary sources for safety demonstration. Consequently, the safety justification for the same item has to be done twice, in two different ways.

This story clearly highlights the gaps between different practices and the gaps between the possible interpretations of the same documents.

4) Conforming to different standards and regulations

Now, EDF wishes to build EPR units in the USA. US authorities provide detailed written regulatory requirements and guidance (contrary to France were only very high level Basic Safety Rules are issued). Also, the standards endorsed by the US authorities are not the IEC standards cited earlier, but IEEE documents. In this case, this is not only a subset of requirements interpretation that will differ but the full content of the provided documents to support the different developments.

Comparing each IEC standard with its approximate relevant IEEE correspondent is difficult, time consuming and does not ensure to have the correct interpretation (the one that will make consensus among all stakeholders) of the different standards.

Difficulties can arise from:

- Vocabulary: terms are not the same. IEC60880 speaks about activities and IEEE1012 considers tasks.
- Semantics: In the end, are we talking about the same thing when using “task” and “activity”?
- Ambiguity: Legal documents and standards contain intended and unintended ambiguity [1], causing interpretation, misunderstanding and negotiation between stakeholders to agree on a common definition.
- Scope: IEC60880 standard covers all the software lifecycle, whereas the IEEEstd1012 focuses only on software validation and verification and needs to be completed with other references.

The key question in this case is to know whether EDF’s EPR solutions can comply with US regulations without having to design a brand new solution.

C. Synthesis

The first story tells us the differences and also the tight relations between standards, which are used as guidelines to fulfill safety requirements. It also partially describes practices that arise from a particular understanding and use of standards. The second story explains the introduction of several practices based on the same reference documents. The last story presents the case of one system that needs to comply with two different sets of reference documents. Figure 1 offers an overview of the evolution of some available references (standards and regulatory documents) regarding several safety related software systems. It also shows how EDF had and still has to deal with them for its projects. A summary of these three stories should highlight the multiple variability dimensions we have to deal with to tackle these questions of standard interpretations in multiple contexts.

III. ON DEFINITIONS, OBJECTIVES AND VARIABILITY

In the following, we define and explain some more terms. We then propose a first global overview of our problem, our objectives and make a particular focus on some of the variability dimensions we highlighted.

A. Definitions

The three more definitions we provide highlight properties on documents or document elements, context elements that will appear while defining our problem.

1) Standard

We adopt the definition of the BSI [2]. A standard is a summary of best practice and is created by bringing together experiences and expertise of stakeholders on a specific topic. It contains a specification designed to be used as a rule, guideline or definition in order to increase reliability, effectiveness and confidence. Standards aim to provide a verifiable and reusable way to answer a problem.

One important point is that standards are designed for voluntary use. However, laws and regulations may refer to standards and make compliance with them mandatory.

There exist several organizations involved in the standardization process like the ANSI, BSI, IEEE, IEC … Regarding nuclear I&C systems, there are mainly two organizations that are the US IEEE’s Nuclear Power Engineering Committee (NPEC), and the IEC subcommittee on Reactor Instrumentation (SC45A). In [6], the author has compared IEEE and IEC standards collections in the domain of nuclear energy. Unfortunately, it is only an attempt to find alignments at the global document level. It does not provide relevant information on specific difference of

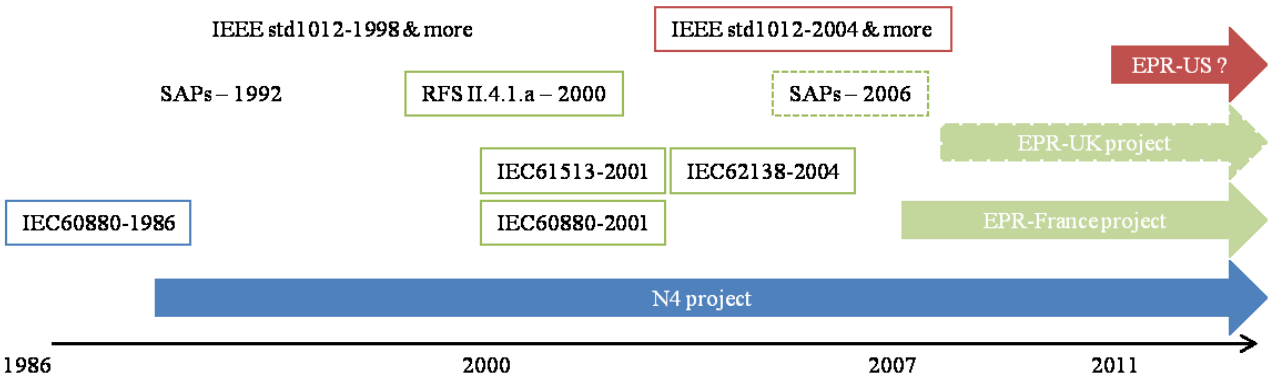

Figure 1 Evolutions of some available references and use during projects

interpretation that may arise between different practices nor provide a fine-grained overview of these alignments/gaps.

The main difficulty is to analyze and compare different interpretations across different scopes and different terminologies. One example of this kind of analysis can be the comparison of the two previously cited standards (IEEEstd1012 and IEC60880).

2) Practice

There are many possible and correct solutions to tackle a problem. Practice is defined as the provided and accepted solution to a specific problem: the way this solution was designed, the complete process it followed (eventually, the compliance to a standard), how it was justified. The main property of this solution is that it had been endorsed by the local nuclear safety authority. As a consequence, several different practices may exist, depending of each country, depending also on past projects.

3) On requirements, recommendation and guidance

Many different definitions exist for a requirement. According to IEEE610.12-1990 a requirement is:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or *possessed by a system or system component to* satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

Many other definitions/decompositions exist: considering for instance functional or non functional properties, specific domain requirements. We have to distinguish here between what is mandatory, what is optional and the range between these two statuses. A requirement is mandatory. A recommendation is optional but should be done. In the standard, there exist three terms (and their synonyms) defining this degree of obligation. "Must" or "shall" define requirements. Recommendations may be determined by the word "should". The word "may" also has another meaning as permission, even semantically weaker than the other statements, and the final conformance evaluation is made regarding these different levels.

There is also a major difference between *regulatory requirements* coming from regulation and *standards requirements* coming from standards. Regulatory requirements are imposed by the nuclear safety authority and one has to comply with if he wants to conform to the law. A standard requirement may become a regulatory requirement when the authority imposes to follow this standard with its own interpretation of the document.

Regulatory guidance is guidance provided by authority and which are not part of the regulatory sources. However, due to the risks and uncertainties entailed by solutions that do not conform to the guidance, regulatory guidelines are considered most of time as requirements.

B. Objectives

The three stories demonstrate that EDF has to:

- follow the evolution of standards (story 1), that represent the best of the state of the art regarding one question and that help to justify to an authority the system' safety;
- understand the interpretation of these documents (standards, guides, regulatory documents) in several contexts (parameterized by time and geographical location) to provide acceptable practices (stories 1 and 2);
- deals with much more than simple requirements to fulfill;
- Know and analyze the impact of documentation changes (stories 1 and 3) in order to provide adapted solutions according to several contexts (story 2).

Actually, all of these aspects remain mainly split between several human experts. As a consequence, one country's practice and its relations with other practices are not formalized, not capitalized and highly dependent from resource management. Worse, no one possesses a complete global view of this knowledge. Considering that these kinds of projects last for decades (many initial years of design, the 40 years of operation and the several years of decommissioning), the question of knowledge management becomes a significant question.

To tackle this issue, we want to propose a domain specific modeling language to represent the variability of the

different practices and their implicit relationships. That way, we will be able to formalize and capitalize experts' knowledge on several different practices and propose a global reference guide for current and future IDF's I&C systems projects. Moreover it will improve traceability aspects by formalizing the existing dependencies between the different project's elements.

C. Variability dimensions

The previous three stories illustrated the need to model multiple, changing elements. According to these stories and in order to start formalizing the problem, we modularize the approach into areas. We describe these areas first. Next section focuses on the dependencies between the elements of these areas and finish by describing a possible example.

Figure 2 summarizes our current understanding of the different elements involved and their relationships. We can distinguish between three main concerns: setting up the corpus of formalized documents, binding these documents to the contexts they are used in and finally the overall project whose aim is to build and qualify a solution relating to the documents used in the different contexts the project wants to address.

We choose to focus in this paper only on the corpus part, its interpretation in particular contexts and the relations they hold.

1) The corpus area

This domain contains information on all referenced documents independently of any context. As said earlier, information can be provided on the type of each document (standard, regulatory text, technical note as exposed before…). Elements can be described in a hierarchical way including containment in sections and subsections. Paragraphs may be typed as definitions, requirements, recommendations or permission using text patterns. References can be captured; information notes can be associated with their parent paragraphs.

2) The interpretation area

Once the document is split and structured in hierarchical typed fragments, the text fragments are eventually interpreted to build a new element, considered as the "operational" one when used in one or several contexts. In this area, a paragraph determined as a recommendation can be considered as a requirement due to its interpretation in a specific context. Another possibility is to have to rewrite the initial text description of a fragment into another one that represents its interpretation. In fact, this domain depicts the contextual practice relating to these documents and the variable part of the document elements.

3) The referential area

We have introduced the notion of context. The context is a pair <country, period> describing the validity of the interpretation: geographically and temporally. Although these fragments remain typed textual fragments and close from the definition of the documentation domain; the major difference, at this level, consists of the semantics added because it now reflects the way they are interpreted within the real context and not only the artificial nature that an algorithm has decided. The referential area contains all these contexts that should be addressed while interpreting documents.

D. About dependencies

We use the term *dependency* to describe the relations between elements of the same area or from different areas. An element depends on another one if it is *refined* by or has an *interaction* with this element [16][24].

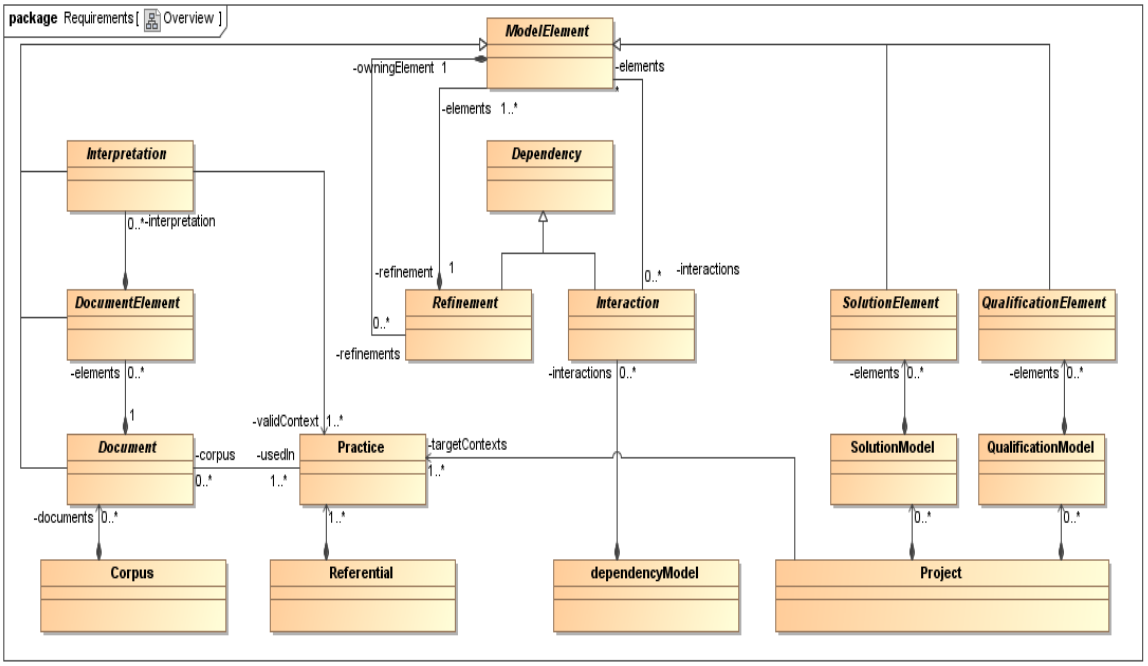

Figure 2: Problem overview

1) Refinements

A refinement may consist of three different actions:

- describing an element in a finer way (total or partial composition);
- detailing some particular aspects of an element (characterization);
- describing different ways to understand, achieve, fulfill an element (specialization)

A refinement is an intra-domain relationship. The refinement of an element into sub elements is the *composition*. A paragraph may contain one main idea (for example, the existence of a precise document) and multiple details concerning partial aspects of this main idea (content of the document) or explaining, for instance, ways to achieve a high level objective without being enough structured to form an independent item. This refinement into a main idea, detailed by other related elements is called *characterization*. Alternatives may be given to fulfill a requirement. This is called *specialization*.

2) Interactions

Interactions are intra and inter areas relationships. They represent the different typed relations we want to represent and will insure traceability between the different elements, provide comparison operators and manage the understanding of changes impacts. Definitions of these relations are following.

"*Conflict*", "*partial equivalence*" and "*total equivalence*" relations express similarity or exclusion between elements. Partial and total equivalence express the facts that at least, elements hold the same general idea, partially or totally, without contradiction. The conflict relation expresses this contradiction between two elements. Obviously, this should be useful while comparing elements of IEC and IEEE standards.

The "*reference*" relation express that an element is citing another one. Explicit and/or implicit cross-referencing is known to be a major issue while trying to understand the context of a requirement and its evolution [20]. This should be useful in case of changes into the referenced element.

The "*require*" relation has a stronger meaning than "*reference*" although they both can be present. An element A requires an element B does not only means that A references B but that B is a mandatory element to achieve A.

The "*coverage*" relation says that two elements A and B have the same interpretation, address the same problem (for example two response time requirements relating to the same element) but if A covers B, then A has stronger constraints or additional properties.

The "*assignment*", "*justification*" and "*qualification*" relations are the specific relations between:

- An element from the documentation domain and an element from the solution domain (Assignment), i.e. this element of solution are constrained in some way by the element from the document domain (e.g. development driven by a standard, function allocation, regulatory requirement).
- An element from the documentation domain and an element from the qualification domain (justification);
- An element from the solution domain and an element from the qualification domain (verification).

E. One example

Let's take an example of what kind of text fragments have to be handled and what kind of analysis can be performed. This example comes from the standard IEC60880.

Chapter 6 of the IEC60880 deals with software requirements and section 6.2 deals with software self-supervision. It contains 6 main text fragments (listed from 6.2.A to 6.2.F).

Fragment 6.2.A is considered as a *requirement* due to the presence of the word shall. It also makes a *reference* to annex A.2.2 section. The following sentence, as it is not in the same paragraph is considered as an *information note* relating to this requirement.

Fragment 6.2.C is considered as a *recommendation* (missing shall and presence of should). Because of the non enumerated category list below, we face a *partial composition* of the requirement into four units.

Fragment 6.2.D is a multiple sentences requirement due to the double presence of shall. It references IEC61513 standard. The second sentence is a characterization of the first one.

Ambiguity is a major actor of this kind of documents as expected to be used in several contexts. "6.2.A" talks about specified time interval. This means one has to decide the exact value of these time interval and has an impact on the final device which will be adopted. "6.2.F" talks about useful diagnostic information. Once again, someone has to decide the exact information.

F. Preliminary discussions

Although the corpus modeling may appear as something mechanical, using specific rules, it could never be perfect. It could be difficult to determine whether a characterization or

6.2 Self-supervision
6.2.A The software of the computer-based system shall supervise the hardware during operation within specified time intervals and the software behaviour (A.2.2).
This is considered to be a primary factor in achieving high overall system reliability.
6.2.B Those parts of the memory that contain code or invariable data shall be monitored to detect unintended changes.
6.2.C The self-supervision should be able to detect to the extent practicable:
- Random failure of hardware components;
- Erroneous behavior of software (e.g. deviations from specified software processing and operating conditions or data corruption);
- Erroneous data transmission between different processing units.
6.2.D If a failure is detected by the software during plant operation, the software shall take appropriate and timely response. Those shall be implemented according to the system reactions required by the specification and to IEC 61513 system design rules.
This may require giving due consideration to avoiding spurious actuation.
6.2.E Self-supervision shall not adversely affect the intended system functions.
6.2.F It should be possible to automatically collect all useful diagnostic information arising from software self-supervision.

a specialization is filled in the text. Distinguishing between a reference and a require relation may be hard to determine.

Moreover, we have to notice that this special formatting is mainly unique to this standard. Standards are written at different times, by different persons, following different typographic rules that make each standard analysis an ad hoc process whose only good property is to provide something reproducible and systematic.

As a consequence, one should be able to check, verify, modify the produced result in order to insure the correctness and the completion of this task. Moreover, traditional properties of elements like maturity, authors, date, contributors or rationales shall be also incorporated [20]. The way to decide the value of these properties may be far from obvious to decide and is part of the interpretation step of the process.

This leaves us mainly with three tasks to perform and related operations:

- Modeling standards by means of refinements modeling and elements typing;
- Modeling variability aspects through the several standards contextualized interpretations of their elements;
- Modeling standards dependencies by providing intra and inter-areas dependencies modeling mechanisms and supporting analysis tools.

In next section, we review several modeling approaches regarding these three tasks and operations.

IV. MODELING REQUIREMENTS DEPENDENCIES

Many fields of research may address these questions of standards requirements, variability management and tasks to perform to formalize practices. Figure 3 presents a roundtrip of all the research domains we are at the crossing of and some interesting keywords we retrieve all along this paper.

While considering requirements formalization and representation, there exist multiple formalism using UML use case diagrams [23], scenarios [14], domain specific modeling languages [1], predicates [9], agent oriented [22] or goal-oriented [12][20] requirements models. Each of them targets a particular objective: specification, analysis, test generation, consistency checking.

A complete survey of these works is not within the scope of this paper and we prefer to focus on some interesting works around two questions. First, the way requirements are modeled and how variability is addressed through these representations. Then, the kind of relationships with the environment these requirements handle. In the following, we first have a glance around one particular goal-oriented approach which is KAOS (Knowledge Acquisition in AutOmated Specification). We then go on with the OMG standard SysML and one of this standard implementation through the Topcased environment. We continue with one particular model-driven approach which is the Unicase environment and finish with the particular question of variability in software product lines and the promising Common Variability Language (CVL) standard.

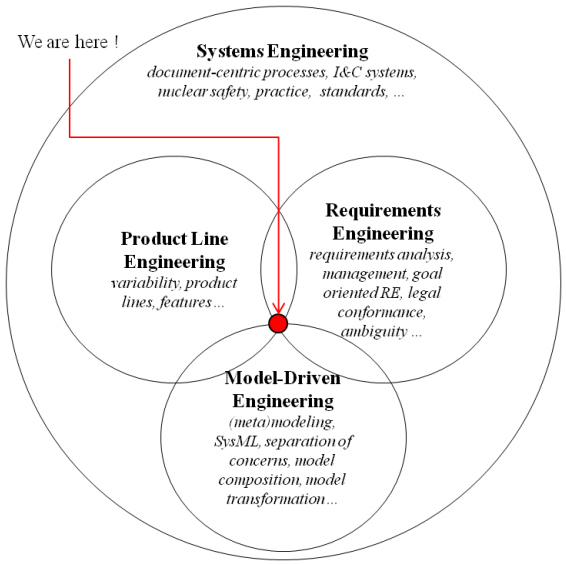

Figure 3 Crosscutting multiple research domains techniques and concerns

A. Goal-Oriented Requirements Engineering using KAOS

1) From goals to requirements

One of the main approaches considers a goal-oriented approach to represent requirements. In Goal-oriented requirements engineering (GORE), goals are main entities for elicitation, elaboration, specification, analysis of requirements. Goals are prescriptive statements of intent whose satisfaction requires cooperation of agents in both software and its environment. Goals and their variability are captured in AND/OR structures which define how they are being refined or abstracted. A requirement is a goal assigned to an agent of the software-to-be. An expectation is a goal assigned to an agent of the environment of the software-to-be. In KAOS [20][21], goals are mainly defined as behavioral goals (clear declaration of an expected behavior) that can be achieved, maintained, avoided or softgoals; whose are not clearly established and are more *satisficed* at different levels than *satisfied* [12]. One should not compare them to functional/non functional requirements which are part of another kind of categorization. This part provides one kind of typing over the different possible goals categories. KAOS has been used over several industrial projects and is fully supported by a complete platform: Objectiver [13].

2) Modeling goals relationships

In KAOS, goals are not only modeled and refined in a sole requirements model. The whole system is represented into multiple dimensions which are intentional, structural, responsibility, functional and behavioral in order to completely cover the WHY-WHAT-WHO questions to the system.

Through these multiple dimensions, goals and later requirements can be involved in relations at several levels [20]. These relations stand at the interface level as:

- Responsibility links between goals and agents,

- Obstruction links between a goal and a obstacle model,
- Concern link, that are relations between goals and object of the object model,
- Operationalization links between the goal model and operation model,
- Coverage links between goal and behavior models.

Concepts and relations in KAOS have to be adapted and specialized to fit with our problem. Document types are missing as it only considers goals. We may consider informally that sections, subsections and such kind of hierarchy could be refined into goals and subgoals. However, it does not comply completely with this notion of packaging. Moreover, many other different items may be refined as goals like requirements and recommendations as they are not actually assignable to any agent. This leads to a problem here as they are already requirements or recommendations. At this moment, goals and subgoals entities merge a lot of very different notions and it does not give a clear representation of the standard. The different range of obligations: requirements, recommendations, informative or normative annexes disappear and must appear in another way. A priority or criticity attribute should not be sufficient to deal with these concepts as they fit more during a development phase for managing purposes.

Apart from the structure, we must also adapt KAOS metamodel to introduce new traceability links such as equivalence, coverage or reference. The conflict notion may be described via a conflict link and a negative participation even though this relationship is between two different model elements. We also believe that standards and regulatory requirements, which are not technical requirements and not designed to go towards development process, cannot be described by only using AND/OR structures. It also lacks elements to model the cross references questions.

B. Modeling multi-facetted aspects using SysML

1) General words on SysML

SysML [17] is a standard language from the OMG, a general-purposes system modeling language extended from UML. The major drawback of UML regarding its usage was that it was felt as to close to software engineering and did not fit for system modeling purposes. The main changes (relevant to our case) are the introduction of new specific diagrams: block diagrams and internal block diagrams replacing the usual UML Class diagram and the requirements diagram we will comment next.

According to the OMG, "a requirement specifies a capability or condition that must (or should) be satisfied. SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements. A requirement can also appear on other diagrams to show its relationship to other modeling elements."

2) On SysML relations

SysML specification includes several requirements relationships, allowing requirements to be refined, derived, satisfied, and verified among other requirements or model elements as shown in Figure 4. Unfortunately, the containment relationship only allows performing a top-down approach on requirements hierarchy. It does not allow a richer composition that may allow feature diagrams for example. The *refine* relationship offers a different approach, while linking a requirement to something described at a finer grain and eventually with a different formalism, for example to map a textual description to a use-case diagram.

Elements reuse, contextual references are addressed across a *master-copy* process, allowing keeping a coherent management regarding requirements evolution. It does not take into account that the textual form in itself may be different from one context to another while the text remains the same. It also does not take into account that elements can be detailed but in a way that each detail is one expected property and not a stand-alone requirement.

The *derive* relationship relates a derived requirement to its source requirement. It appears to look like the merged interpretation of the characterization and specialization dependencies we have defined earlier. Another mechanism provided is to fill requirements matrix which deals with all requirements derivations. Both relation and matrix insure traceability aspect of the requirements engineering process.

The *satisfy* relationship describes how a model element satisfies one or more requirements. It's the equivalent of our allocation relationship. It holds an operational point of view whereas our allocation relation is an abstract relation saying: This requirement is taken into account in this solution element and reversely, this element of solution provides an acceptable way to fulfill this requirement.

The *verify* relationship defines how a test case or other model element verifies a requirement. It fits to our definition of the qualification relation. The specification considers the test-case as any generic mean to provide verification as we are.

C. Modeling requirements using model-driven engineering tools

1) Modeling requirements using Topcased

(Toolkit in OPen-source for Critical Applications and SystEms Development) Topcased [18] is an eclipse based

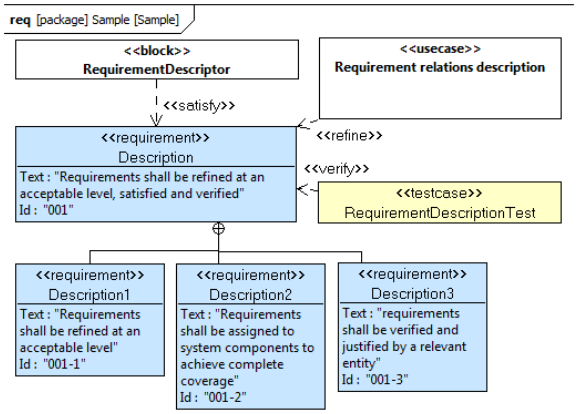

Figure 4: SysML requirements relations

environment designed for critical systems. The fact was that design of critical systems requires the mix of many different tools over the system life-cycle. These tools targeted several and different aspects of the project and, in this case, for the modeling aspects. The idea behind Topcased was to offer one unique Computer Aided Software Engineering tool that can offer functionalities like generating, analyzing, transforming these different models, while enhancing traceability and finally improving system robustness.

Interesting functionalities of Topcased are one implementation of a SysML metamodel and the possibility to import requirements directly from existing documents (with respect to some specific document formatting). Via regular expressions analysis, formatting rules, it is possible to decompose a document into a hierarchical set of requirements. Unfortunately, source documents must have a too constraining formatting and a specific requirement tagging policy that is not present in the documents we wanted to analyze, documents existing over different formatting. Moreover, it does not allow a complete typing of our elements in different elements (requirements, recommendations, definitions, annexes, and notes as for the documentation domain).

2) Modeling requirements using Unicase

Unicase [19] is an eclipse-based framework that offers a navigable and unified structure based on Unified Requirements Modeling Language (URML) [5]. Unicase allows linking scenarios with functional, non functional requirements, use cases and other model elements (customizable as it is based on EMF and Ecore) with standard properties (unique ID, description, priority, cost). Unfortunately, it focuses in navigation which is useful to follow evolution of referenced elements, manage requirements during the project life (assigning actions on the requirements) and enhance automatic traceability with a project development perspective. It does not take into account the ambiguity (managed as defect that should be fixed) or the several different interpretations possible that are context valid [7].

D. CVL and software product lines engineering

1) Variability in product lines

Product lines engineering aims to improve reusability within a family of related systems. This family can be defined as a set of common elements, named core assets and variable elements [10] [16]. One of the key tasks being the identification and separation of what is common and what is variable. In fact, it may be a similar task as to compare two standards' elements. At the same time, the other key task managing adaptation of these variable elements by orchestrating the possible configurations.

Feature diagrams are used as representation language for this variability concern. Introduced by Kang [8], they offer a compact but exhaustive way to represent configurations of a product line. These configurations are expressed in term of features and how they can be composed, following a parent-child relationship from the more abstract concept to the more concrete one.

2) About Common Variability Language (CVL)

Concerning the sole variability question, we way notice the current request for proposal from the OML around a Common Variability Language [3]. CVL's objectives are to enable the specification of the variability in product line models in order to support product line modeling. One of the main concepts is also to promote division of labor and separate design concerns form variability concerns. This idea of weaving variability can also be found in [11] but in a different way as it promotes fusion of both concerns instead if separation. Thus, CVL will be based on four models. The *base model* is a representation of the problem without variability. The *variability model* where will be expressed variability. The *resolution model* will store the user's choices regarding the configuration from the variability model. Finally, the *resolved model* will embed the derivation of the base model and the variability model, producing a final model conforming to the base metamodel.

CVL address the question of variability at a very general purpose level, separating design time from variability analysis. Both CVL and [11] approaches should be promising to fit to our process by modeling our documents first (some kind of base model) and then, bringing the variability aspects.

E. Synthesis

Within the perspective of the three tasks to perform and their related operations highlighted in section III.F, we summarize the previous analysis in Table 1. In this table, a smiling face is for a concern which is taken into account with a good expressivity (even if it doesn't correspond to our needs). A neutral face is used for concerns partially taken into account or without enough expressivity. A sad face is used for concerns not taken into account or too poorly.

Actually, the majority of modeling tools target only one current context. They do not aim to deal with multiple contextualized interpretations. They are also badly designed to model intra-area dependencies as they do consider or aim

Table 1: Approaches' fitness on to three tasks and their operations

Task	Operations	KAOS	SysML	TopCased	Unicase
Modeling standards	multi-document partitioning	☹	☹	☺	☹
	Typing	😐	😐	😐	😐
Modeling standards interpretation variability	Contextualized interpretation	☹	☹	☹	☹
Modeling standards dependencies	Intra-area dependencies	☹	☹	☹	☹
	Inter-area dependencies	☺	☺	☺	☺
	Elements alignment	☹	☹	☹	☹

to build requirements as non ambiguous, precise, clear artifacts. These two points represent the field of variability for our concerns. They do not take into account the different elements types we can find in standards (requirements, recommendations, annexes, definitions and so on) as they consider only requirements even if customization is still possible (using stereotypes, regeneration of the domain metamodel or attribute properties). They take into account inter-area depencies by using satisfying/verifying like relationships whereas we consider more relations have to be defined for comparison purposes.

V. DISCUSSION AND FUTURE WORKS

In this paper, we have defined some areas of our problem and dependencies that should have to be expressed regarding the formalization of standards, their possible interpretations and their relationships. With these elements defined, we have reviewed some modeling approaches and identify their limits in front of what we expect to do and the gaps we have to fulfill.

Once processed, results of a standard parsing phase can produce model elements, which can then be enhanced with their different interpretations (variability aspect) and dependencies (traceability and analysis).

None, to the best of our knowledge, have tried to leverage model-driven engineering techniques to represent this knowledge as model elements and consider the ambiguity of documents this way. Here ambiguity is something natural that has to be modeled and manipulated and not fought. Formalizing ambiguous requirements and providing some of their possible interpretations are a way to make them clearer, more sharable and reusable.

All of the approaches lacks of expressiveness regarding typing or dependency links and are currently software development-centered. Actually, works are driven by the objective to build or use technical requirements, something that could later be addressed with a program. In our context, we are knowledge-centered. We want to describe several interpretations of the same element with rich expressivity, give the opportunity to analyze them, to synthesize and capitalize this knowledge using models. Providing a DSML for these variability and traceability questions and building this knowledge model will be the major contributions of this work where software is just a part a very larger whole and also has to be abstracted.

REFERENCES

[1] Baudry B., nebut C., Le Traon Y., "Model-driven Engineering for Requirements Analysis", In EDOC'07 (Entreprise Distributed Object Computing Conference), (2007)

[2] British Standards Institutions: http://www.bsigroup.com/

[3] CVL: http://www.omgwiki.org/variability/

[4] French Nuclear Safety: Development and utilisation of probabilistic safety assessments, (2002)

[5] Helming, J., Koegel M., Schneider F., Haeger, M., Kaminski C., Bruegge B., Berenbach B., "Towards a unified Requirements Modeling Language", In Proc. Work. REV2010 Page(s): 53–57. (2010)

[6] Johnson, G., "Comparison of IEC and IEEE standards for computer-based control systems important to safety". Nuclear Science Symposium Conference Record, (2001)

[7] Kamsties E., "Understanding Ambiguity in Requirements Engineering", In Aybüke Aurum & Claes Wohlin, editors, Engineering and Managing Software Requirements, chapter 11, pages 245-266. Springer (2005).

[8] Kang K. C. , Cohen S. G., Hess J. A., Novak W. E., Peterson A. S., "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Technical Report, (1990)

[9] Maxwell J. C., Anton A. I., "Developing Production Rule Models to Aid in Acquiring Requirements from Legal Texts", In Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, (RE '09). IEEE Computer Society, Washington, DC, USA, 101-110.

[10] Moon M., Yeom K., Chae H. S., "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line", IEEE Trans. Softw. Eng. 31, 7, 551-569, (July 2005)

[11] Morin B., Perrouin G., Lahire P., Barais O., Vanwormhoudt G., Jézéquel J-M., "Weaving Variability into Domain Metamodels", MoDELS 2009, pp 690-705, (2009)

[12] Mylopoulos J, Chung L., Yu E, "From object-oriented to goal-oriented requirements analysis", Commun. ACM 42, 1, 31-37. (January 1999)

[13] "Objectiver: http://www.objectiver.com

[14] Rolland C., Souveyet C., Ben Achour C., "Guiding Goal Modeling Using Scenarios", IEEE Trans. Softw. Eng 24(12): 1055-1071 (1998)

[15] Safety Assessment Principles for Nuclear Facilities 2006 Edition, Revision 1

[16] Schobbens P.-Y., Heymans P., Trigaux J.-C, Bontemps Y., "Generic semantics of feature diagrams", Comput. Netw. 51, 2, 456-479. (February 2007)

[17] SysML: "OMG Systems Modeling Language (OMG SysML), Version 1.2," jun. 2010. (2010)

[18] TopCased : http://www.topcased.org/

[19] Unicase: https://teambruegge.informatik.tu-muenchen.de/groups/unicase/

[20] Van Lamsweerde A., "Requirements engineering: From System Goals to UML Models to Software Specifications". Wiley (2009)

[21] Van Lamsweerde A., "Requirements engineering: from craft to discipline", In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering (SIGSOFT'08/FSE-16). ACM, New York, NY, USA, 238-249. (2008)

[22] Yu E., "Social Modeling and i*", In Lecture Notes in Computer Science, Volume 5600, Conceptual Modeling: Foundations and Applications, Pages 99-121 (2009)

[23] Yue, T., Briand, L.C., Labiche, Y., "A Use Case Modeling Approach to Facilitate the Transition Towards Analysis Models: Concepts and Empirical Evaluation", In Proc. 12th Int. Conference on Model Driven Engineering Languages and Systems (MODELS '09), pp. 484-498, (2009)

[24] Zhang W., Mei H., Zhao H., "A Feature-Oriented Approach to Modeling Requirements Dependencies", In Proc 13th IEEE Int Conf on Requirements Engineering, pp 273-284. (2005)