# The DODT Tool Applied to Sub-Sea Software

Tor Stålhane
Dept. of Computer and Information Science
Norwegian University of Science and technology
Trondheim, Norway
stalhane@idi.ntnu.no

Tormod Wien
Corporate Research
ABB
Oslo, Norway
tormod.wien@no.abb.co

***Abstract*–Using natural language is still a common form of writing software requirements. Tools and techniques to improve the quality of natural language requirements may give better results than attempts to convince industry to use something else. We have combined natural language requirements with tool support using boilerplates and domain ontologies, enabling detection of ambiguities and incompleteness in requirements. This paper reports on a case study where requirement analysts used the developed tool to analyse requirements for a safety-critical control system. The experience showed that people were able to use the tool to develop a domain ontology and apply boilerplates to describe requirements in a structured way, yielding requirements readable for humans and analysable for the tool. The tool support improved the quality of requirements by reducing ambiguities and inconsistent use of terminology, removing redundant requirements, and improving partial and unclear requirements.**

***Index Terms*—requirements specification, natural language, patterns, ontologies**

## I. Introduction

When developing a software system, the requirements are the basis for all later work. If the requirements are wrong, imprecise or incomplete, the same will hold for the final system. On the other hand, if the requirements are complete, consistent and unambiguous, the final system will usually have a high quality. More than 40% of all problems in a typical software project may be related to requirements [1]. The main reason for this is that the developers have nothing, except the users and applicable standards to check the requirements against. Later in the process it is possible to compare the design against the requirements, the code against the design and so on, to make sure that nothing that is needed is lost and nothing that is not needed is added.

Since the requirements are an important communication medium, they need to be understood by all stakeholders. Thus, we have chosen natural language (NL) as a means of specifying requirements. If we want to use tools to formally check the requirements' quality, we will need a formal NL presentation. We have selected the boilerplates representation to achieve this.

The rest of this paper is organized as follows: Section II gives a short background while section III discusses related work. Section IV gives an overview of the research methodology used and presents the research the questions while Section V shows our analysis of ABB's experiences with the tool and thus the answers to the three research questions. The paper is rounded off with Section VI which presents our conclusions and Section VII where we discuss where to go from here.

## II. Background

### *A. The DODT Tool*

The methods and tool discussed in this paper all stem from the CESAR project – Cost-efficient methods and processes for safety relevant embedded systems – which is a European funded project from ARTEMIS Joint Undertaking (JU). The project had requirements handling as one of its top priority items. To achieve this, the project developed the DODT (Domain Ontology Design Tool) for requirements analysis. This tool is a further development of the GNLQ (Guided Natural Language) tool which is now available via SourceForge. Even though this approach helped requirement engineers to develop better requirements, we were not able to attack important quality issues.

One of main goals for the new DODT tool is the ability to check requirements for completeness, consistency and unambiguity, based on information in domain ontologies. For the DODT tool, completeness means "covers all relevant elements in the system ontology used".

We need two concepts to describe the DODT – ontologies, and boilerplates [2]. They interact with the tool as shown in Fig 1. One of the first to use boilerplates for software requirements were Hull and Dick [3]. The main reason for introducing boilerplates was to add structure to the NL requirements. A general requirement formulated using a boilerplate has three parts [3] – a precondition, an active part and an action-condition. We use the "<" and ">" to isolate the variable parts of the boilerplate requirement.

- Pre-condition – one or more operational conditions that need to be true in order to move on to the action part of the requirement, e.g. "While <state>…" or "If <event>…"
- Active part - one or more actions performed or abilities made available if the pre-condition is true, e.g. "…<system> shall <action>" or" …<system> shall allow <entity> to be <state> "
- Action-condition – one or more operational conditions or restrictions that apply to the action or the way the action is performed, e.g. "...without <action>" or " ...within <number> <unit>"

For a boilerplate, an operational condition is either the occurrence of an event or that the system is in a defined state. Using boilerplates we get a set of semi-formal requirements. These requirements are both easy to read and possible to handle by software tools. An example of a complete boilerplate requirement can be written as shown below.

If <event>, <system> shall <action> within <number> <unit>.

This template can be used to write the following requirement: **If <temperature reaches max temperature>, <boiler controller> shall <reduce power to heating unit> within <2> <seconds>.**

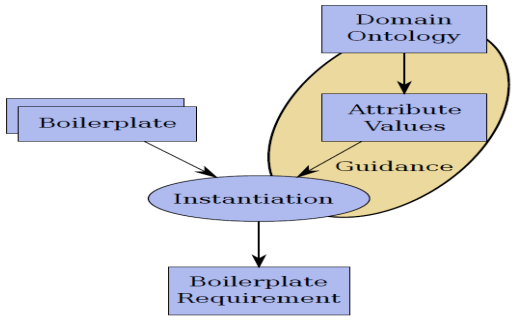


Fig. 1 DODT overview

Domain ontologies make the tool more than just a database for requirements [4]. We can make sure that similar requirements are written using the same structure and that each term is interpreted correctly. If we go back to the example above, there is no doubt as to how <2> <seconds> shall be interpreted since it is preceded by the term **within**.

Ontologies have three parts – a thesaurus, a set of relationships between terms in the thesaurus and rules for making interferences based on these relationships. The information needed to build ontologies is in our case mainly found in relevant standards and glossaries. Building ontologies are at the present not an automatic process. Fig. 2 shows part of a steam boiler controller otology

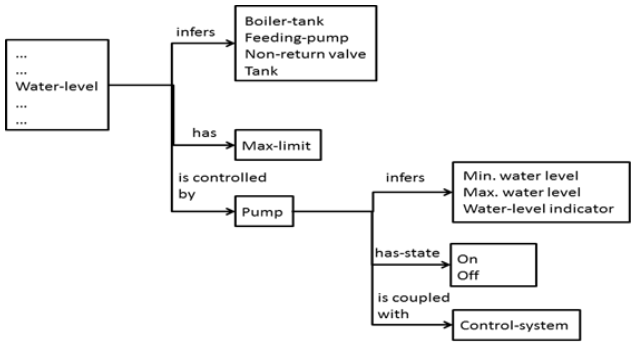


Fig. 2 Part of a steam-boiler ontology

The DODT uses ontologies for three purposes:

- To check requirement quality.
- To handle synonyms. E.g. in a steam-boiler ontology, the terms tank, vessel and boiler-tank will be synonyms.
- For safety analysis. Each ontology entity linked to a physical unit or a system component has a set of generic failure modes.

In the DODT tool, requirement quality is defined by the following metrics:

- Completeness: Check for missing requirements. The analysis reports all terms that are linked via relations from requirement terms and that do not occur among the requirements
- Inconsistency: Terms occurring in the requirements and linked by a contradicting axiom or requirements containing identical sets of terms and exactly one cardinal which is different in the two requirements.
- Ambiguity: Using terms that have multiple interpretations despite the reader's knowledge of the context. It does not matter whether the author unintentionally introduced the ambiguity, but knows what was meant, or he or she intentionally introduced the ambiguity to include all possible interpretations
- Noise: Terms not contained in the domain ontology
- Opacity: Unrelated terms in a requirement which possibly means that a concept is used erroneously
- Similarity: Requirement pairs that is similar by sharing a configurable number of concepts. Alternatively the analysis is also able to detect requirement pairs that use exactly the same set of terms
- Obsoleteness: Requirements using obsolete ontology concepts

The DODT tool handles ambiguities by looking for a term with the same meaning which is already in the ontology. If another term is defined in the ontology to precisely represent the intended concept the term from the ontology should be used instead. If the less precise term is defined as a possible synonym to the preferred term, the tool could be able to propose this change to the analyst.

The DODT tool handles completeness by ensuring that all ontology components are referred to in one or more requirements. If we consider the ontology shown in Fig. 2, we see that if we have a requirement which mention the term "water level", the tool will require that we also have requirements that refer to the boiler-tank, a feeding pump and a non-return valve. Otherwise, the tool will indicate that you have one or more missing requirements.

### *B. Case Study Context*

ABB has already decided to use the DODT tool to improve requirements handling, especially to reduce ambiguity, improve communication and understanding and avoid unclear and redundant requirements. ABB therefore wanted to assess the tool's supportiveness when it comes to removing ambiguous and redundant requirements, different term used to describe the same "thing", unclear or ambiguous terms and

partial or unclear requirements. ABB's needs are covered by the part of the DODT tool's problem identification areas "inconsistency" and "ambiguity".

## III. Related Work

### A. Use of NL in Requirements

There has been a lot of work done on the use of NL in requirements specification – see the work of Fabbrini et al. [5] which sums up some of the most important contributions. It is possible to use unconstrained natural language and then apply a tool to check for inconsistencies, ambiguities and so on. A case study performed by Fabbrini et al. shows promising results in that the tool QuARS was able to identify several common problems such as vague and underspecified requirements [6]. For the 444 requirements for a space application written in NL they found 257 problems, and among these there were 17 vague requirements and 48 requirements that were underspecified.

The EARS templates (Easy Approach to Requirements Syntax), which has been developed by Rolls Royce [7], and the later version BIG EARS [22] have many ideas in common with boilerplates. Just as with boilerplates, EARS allows the user to combine templates to create more complex requirements than can be done with each of the templates alone. However, they warn people that the strong side of EARS – its simplicity – easily can be compromised by creating too complex requirements. As an alternative, they recommend using truth tables when the requirement pre-conditions get overly complex. A case study on the use of EARS, reported in [7] shows that compared to unstructured text, EARS give requirements that are easier to test and use as a basis for implementation, has less duplicates, omissions and complexity and the requirements are less ambiguous and vague. EARS has just two weaknesses: (1) it has no tool support and (2) since it does not contain domain knowledge, it is not able to check for requirement quality factors such as completeness, consistency and ambiguity.

SPS is a pattern-based, restricted English grammar [8]. It is structured as a set of logic propositions rather than as NL statements. For instance: "Globally it is always the case that if P holds, then S holds after at most c time units". Here P and S are Boolean proposition values and c is an integer. The SPS is assessed through a case study by A. Post et al. [9]. The case study showed that only 25 out of 250 requirements used in the case could *not* be expressed using SPS.

### B. Use of Ontologies in Requirements

One of the languages that have been used for requirements is PSL – Process Specification Language – see Gruinger and Menzel [10]. The tool uses ontologies but only to define the requirements structure. No domain ontology is involved. Bloem et al. have constructed the tool RATSY using this language [11]. The tool can help software engineers to develop a property based design based on the PSL specification. The weak side of this approach, beside the lack of a domain ontology, is that the resulting specification is just as complex as the code, there is no NL support and no domain knowledge included.

Siegemund et al. [12] give an overview of the challenges and opportunities when using ontologies in requirements engineering. The authors focus on how an ontology can help the requirements engineer to identify e.g., consistency and completeness. The ODRE (Ontology-Driven Requirement Engineering) meta-model for requirements, presented in [12] only contains a requirement ontology, not a domain ontology. Falbo et al. [13] has also introduced a requirement ontology, based on the unified foundational ontology – see [14]. Again, this is a requirements ontology and not a domain ontology. The same is true for Innab et al.'s work [15]. Thus, these methods are of limited use to the working requirement engineer.

Two papers – one by Castaneda et al. [16], and one by Kaiya and Saeki [17] – both apply domain ontologies and requirements ontologies in requirements engineering. None of them have developed a tool but are able to demonstrate that their approaches will help to identify requirements problems such as incompleteness and inconsistencies.

No tool or approach that we have identified (1) allows NL requirements, (2) has a set of templates to structure the NL requirements and to enable requirements analysis and (3) has a ***domain*** ontology which can be used to analyse the requirements' quality. The DODT, however, meets all of these needs.

## IV. Research Methodology

### A. Research Questions

To see if the DODT can fulfil the needs identified in section III and to evaluate the DODT's usefulness to ABB, we need answers to three research questions:

- RQ1: Can we move requirements from Excel or Word to DODT without losing important information? See section V.A.
- RQ2: Can we use DODT to build a useful ontology based on free text requirements and applicable standards? See sections V.B and V.C
- RQ3: Will the DODT help the engineers to develop a requirement set that is consistent and unambiguous? See section V.D

### B. Research Design and Operation

For the evaluation ABB selected the requirements specification originating from the ISO 13628-6 [18] standard for subsea installations. These requirements were reformulated and new requirements were added to take into consideration customer requirements and ABB internal requirements and constraints. This resulted in 97 system specific requirements, which were the starting point for our evaluation process. In order to answer the three research questions we performed the relevant project activities and collected qualitative and quantitative data.

The requirements were originally on a tabular format comprising identifier, headline, requirement text, motivation for the requirement, the source of the requirement, e.g. internal

(ABB) or a standard, customer and priority. The requirements were converted to boilerplates and stored into the DODT tool as boilerplate requirements – see Fig. 1.

In order to transform the requirements in the DODT tool from a natural language presentation to boilerplates the RE used the process shown below. Note the tight coupling between the requirements insertion process and the ontology development process.

- Build the ontology based on the NL requirements and the relevant standards
- Step through the requirements one by one, rewriting when necessary, and adding concepts to the ontology when missing concepts are encountered. The tool will give assistance here but the selection of the appropriate boilerplate is up to the requirements engineer.
- Use the requirements analysis feature of the tool to search for incomplete, unclear or ambiguous requirements
- Create a list of unclear concepts and/or requirements that couldn't be resolved on the fly. This list is sent to the subsea systems team for clarification.
- Transform selected requirements into boilerplates

As mentioned earlier, ABB's main goals were to identify inconsistencies and ambiguities in the requirements. This was done using the DODT's requirements analysis capabilities.

### *C. Validity Evaluation*

In general we need to consider three groups of validity threats: Threats to conclusion validity, threats to construct validity and threats to external validity. Since this is a case study and not an experiment, issues related to construct validity are not present.

Internal validity – did the observed effect really stem from the treatment, in this case, from the use of the DODT? Since the observed positive effects are related to the ontology and to a partly automation of requirements handling – see section VI – we are confident that the observed effects stem from the use of the DODT.

Conclusion validity is concerned with data quality. The qualitative data collected during this case study were the RE's experiences of using the tool. The quantitative data were collected from documents generated during the requirements process. In both cases the data are representative for the engineer's use of the DODT on the requirements for the system under consideration.

External validity is concerned with generalizations of the results, both within the company and to other companies in the same application domain. Development of software for a subsea installation is no different from the development of other safety-critical software considering IEC 61508 or other, similar standards – e.g. ISO 26262. We thus see no threats to the conclusion validity.

## V. RESULTS AND ANALYSIS

### *A. Requirements from Word and Excel to DODT*

Before using DODT, all requirements were written in Excel spread sheets or in Word documents. A typical example from the Subsea Requirements Specification is shown in Table 1. ABB wanted to preserve this information when importing the requirement into DODT. Since the DODT requirement format supports all fields in the table above – see Fig. 3 below - it was a simple task to map the requirements from Table 1 to similar fields in the DODT requirements format. The requirements in the Word document were imported into Excel. We then used an Excel macro to format each requirement according to the column headers as shown above and then import the Excel tables into DODT. The final result of this process was 97 requirements in the DODT tool.

Table 1 ABB example requirement format

| **ID** | **Headline** | **Pri** |
|---|---|---|
| **DLV-01.03.01** | SUBSEA FACTORY SYSTEM SIZE | 1 |
| **Requirement**<br>The ABB subsea control system shall be configurable to control and monitor a subsea factory consisting of :<br>- Multiple well templates<br>- Multiple separation and boosting processes<br>- Subsea power electronics for distribution and conversion | | |
| **Motivation**<br>Typical contents of an installation according to Market Case 3.<br>NOTE: It is assumed that a subsea factory is split into several template clusters separated up to 50km, and that each template cluster require several control nodes. | | |
| **Source ABB**<br>ABB | | |

RQ1 can thus be answered in the affirmative – we can move requirements from Word documents and Excel spread sheets to DODT without losing information.

### *B. Converting Requirements to Boilerplate Representation*

The basis for the DODT [19] is the boilerplates. Boilerplates as a means for software requirements specification were introduced by Hull and Dick [3] and are a set of requirements templates. The main reason for introducing boilerplates was to add structure to the NL requirements. In this way, there is a limited set of ways to formulate a requirement which again helps us to ensure a consistent representation. The CESAR project has developed this concept

further and the current set of requirement boilerplates has 34 items and is used by e.g., Infineon, Airbus (G) and ABB – CR.

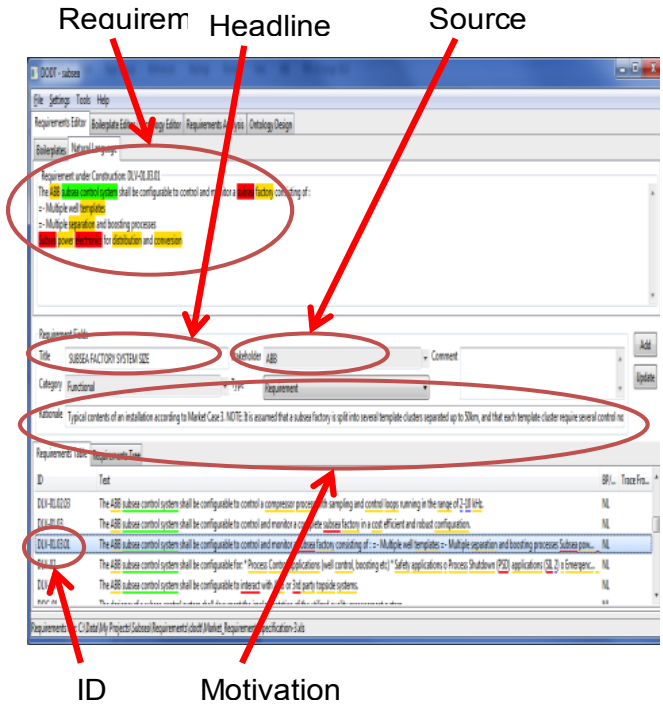


*Fig. 3 ABB subsea requirements imported into DODT*

When we started this work, no ontology existed for the subsea equipment domain. As we have observed in the CESAR project [20] it is difficult to build a complete ontology from scratch. The ontology was thus built step by step – starting with the abbreviations and definitions used in the originating requirements specification, based in ISO 13628-6 [18] and on concepts used in each requirement – adding new concepts, relations and axioms as the needs arose. The ontology was developed using the process described in section IV B.

We started the process described in IV.B with the 40 most used concepts. During the process, four concepts were removed, six concepts were changed and 18 new concepts were added. The final ontology contained 54 concepts.

When a requirement is entered into the DODT tool, all nouns are marked as green (found in the ontology) or red (not found in the ontology. This information eases the work of defining and adding new concepts to the ontology.

The task of creating an ontology must carefully consider every proposed concept to decide whether it belongs to the specific domain. As an example, consider the following requirement:

***"The SCPS shall have an architecture and system design which enables maintenance and replacement of parts of the system without affecting the rest of the installation."***

In the example the term SCPS (Subsea Control and Protection System) will be part of a "subsea ontology" while the other terms are generic.

The answer to RQ2 – Can we build a useful ontology based on free text requirements and applicable standards – has two parts:

- It is possible to build the main structure of an ontology based on the standards applicable to the domain in question.
- This ontology must be extended and refined as we insert new requirements into the DODT tool.

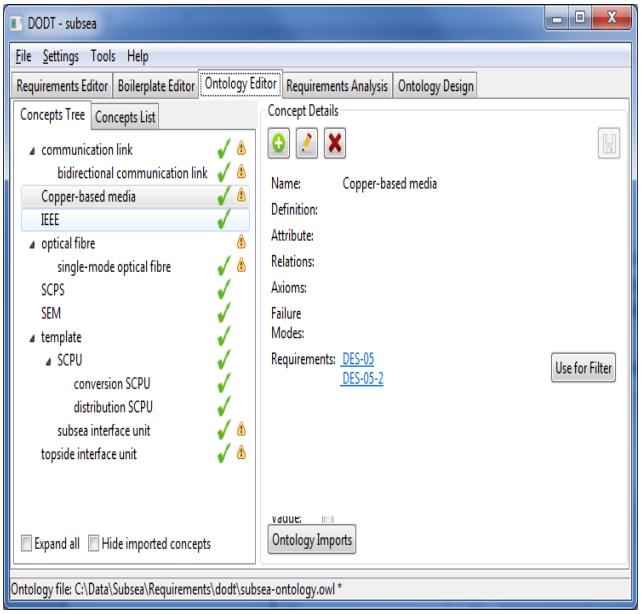


Fig.4 Ontology screenshot

Thus, we can use the standards to give us a good starting point but building a complete ontology for a domain is an iterative process.

### *C. Requirements Analysis*

The requirements analysis feature in the tool is important to ABB. Fig. 5 shows the metrics view for the first version of the requirements – see Section II for definitions. We see that the tool found 96% noisy terms in the original set of requirements, i.e., terms that are not defined as concepts in the ontology. It also indicates that there were 15% ambiguous requirements and that 69% of the requirements have unrelated concept pairs used within one requirement (Opacity). The individual tabs provide a mean to go into each requirement in details and to decide if a new concept should be added to the ontology, if the term should be changed to a concept already defined, i.e. rewrite the requirement, or if it should just be ignored.

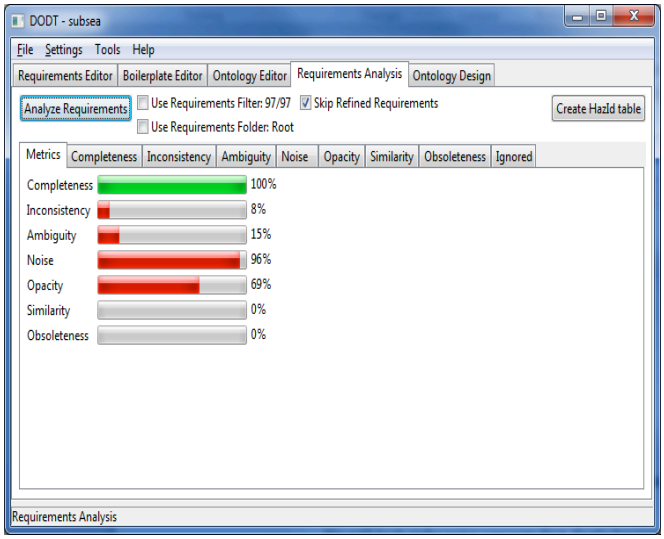


Fig. 5 DODT Metrics screenshot

The requirement engineer studied the noisy concepts first, firstly because they in most cases are simple to handle and secondly because they are important for the ontology enhancement. The DODT can show all noisy concepts – see Fig. 6. Some of the terms are obvious noise terms – e.g., ABB and note – and should be removed. Other terms are needed and must be inserted into the ontology – e.g., canister and Ethernet.

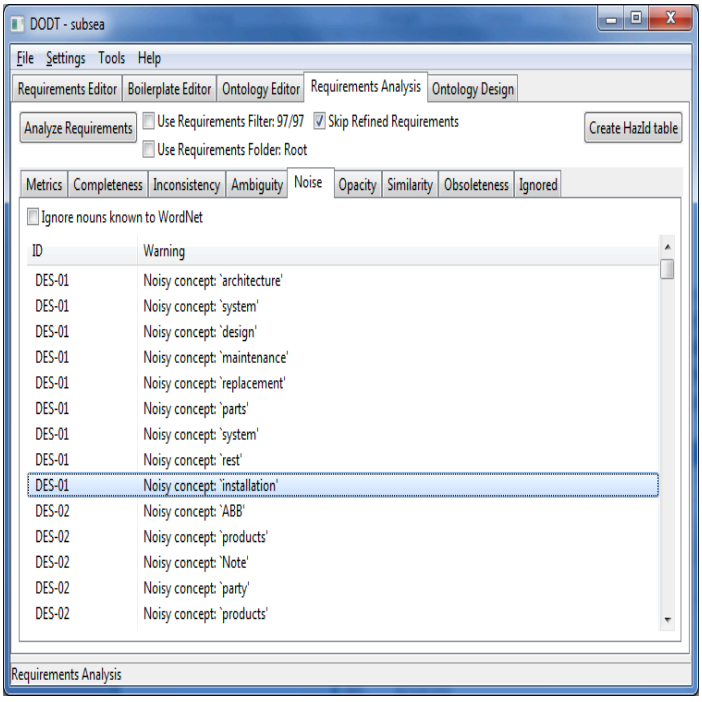


Fig. 6 Noisy concepts

*D. Using the Ontology –Ambiguity and Opacity*

1) *Ambiguity:* As can be seen in Fig 5, 15% of the requirements are ambiguous. The screen shot in Fig. 7 shows the ambiguous requirements – for instance requirements with the ambiguous concept communication link. The reason for this being ambiguous is that the requirements engineer has defined specialized subclasses of this concept. It is now possible to look into the requirements where this concept is used in order to search for a resolution. From the screenshot in Fig. 7 we see that the concept communication link is used in the requirement DES-06:

"There shall exist a secondary communication link between the SCPS and the topside facility, either over single-mode optical fibre or on the power lines".

This requirement refers to a communication link that can be realized using either optical fibres or power-line modems. In this case the requirements engineer therefore selects to keep the generic concept. In the tool this can be done by marking the requirement as "ignored" in the ambiguity tab, meaning that it will not be taken into consideration in the subsequent analysis. The next communication link concept is in requirement DLV-01.01.01:

"The SCPU shall include…Redundant communication link over Ethernet TCP/IP to topside and/or subsea control system(s)"

The requirement requires a redundant communication link, but it doesn't say anything about the physical media. Is it fibre optic, power-line modem or maybe both? What is meant by the term "redundant"? This requirement goes into the list of terms that need subsea system group clarification.

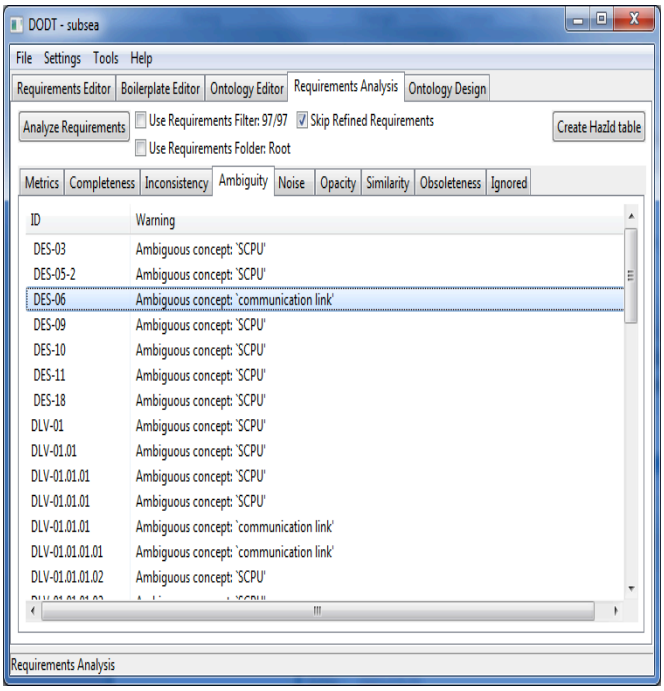


Fig. 7 Ambiguous requirements

We can go through all items in the list of requirements comprising ambiguous concepts. The requirements engineer is

offered three options: (1) edit the requirement, (2) ignore it, i.e., keep as is, or (3) replace it with one of the subclass concepts. The ambiguity view offers an efficient way of going through requirements and either keeping it in places where appropriate or replacing the concept with a more precise subclass concept.

2) *Opacity:* The first iteration with just a few concepts, also found some concepts marked as unrelated, meaning that the concept may be used erroneously in the requirement. Fig. 8 shows the tool's screen for opacity. Fig. 5 shows that at the start of the process there is 66% opacity – e.g., SCPU together with template and SCPU together with SCPS. One of the requirements contains the two unrelated concepts template and subsea interface unit:

"Templates shall be interconnected (via subsea interface units) with redundant, bidirectional fibre-optics, and communicate over Ethernet TCP/IP".

The question is if there is a relation between the two concepts. The definition of *template* shows that this is *a large steel structure which is used as a base for subsea structures such as…*etc. That is, a *template* is a structure for holding all physical entities related to a subsea installation.

It is possible to extend the *template* definition with the axiom *contain*, so that the *template <contain> subsea interface units* (and other units). Fig. 9 shows the new ontology, where the concept tree is extended. The concepts *SCPU* and *subsea interface unit* are now *contained* in the template.

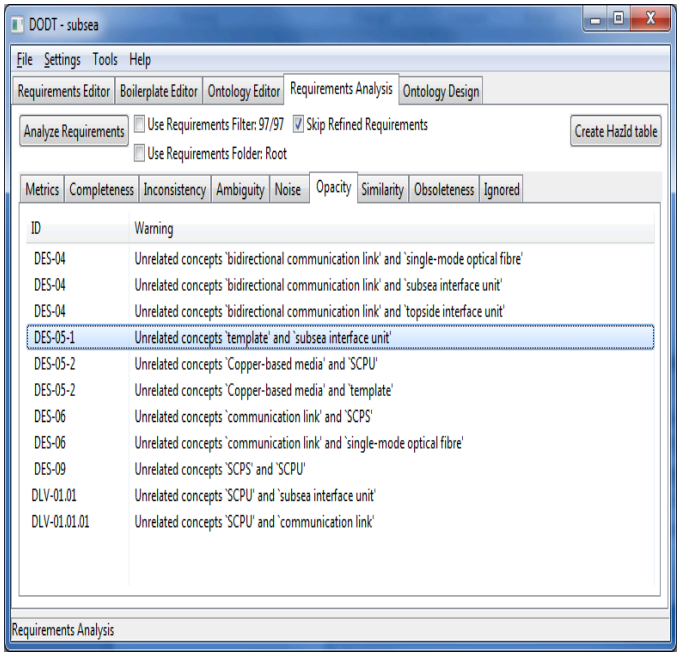


Fig. 8 DODT unrelated concepts

If the requirements analysis is run again, those requirements disappear from the opacity table. The DODT requirements analysis feature helps us to identify opacities and enables us to relate new concepts, thus further contributed to clearly and precisely written requirements.

Of the 97 requirements that we started with, 35 requirements have undergone minor changes, e.g. changed one or more terms in order to comply with the ontology. Ten requirements have been completely rewritten for the same reason. In the end, this left us with 90 requirements. Thus 50% of all requirements were changed during the requirements analysis process.

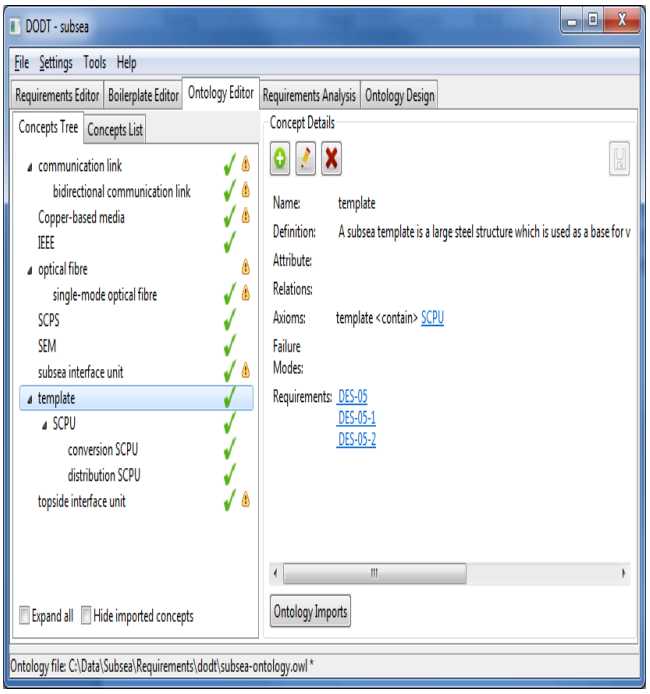


Fig. 9 DODT concept definition example

We will conclude that the DODT tool help the requirements engineers to identify ambiguities and opacities. Thus, RQ3 is answered in the affirmative.

## VI. Conclusions

The final result of the requirements conversion process was a complete set of requirements in DODT. Thus, RQ1 can be answered in the affirmative – we can move requirements from Word and Excel to DODT without losing important information. When it comes to R2, the answer is that yes, we can develop an ontology based on applicable standard and free text requirements but only in an iterative way – start with an ontology based on applicable standards, add requirements, adjust the ontology, add more requirements and so on.

The answer to RQ3: "Will the DODT tool help the engineers to develop a requirement set that is consistent and unambiguous" is definitively "Yes".

We have also seen that the tool has the potential to improve communications – in our case between requirements engineers and domain specialists. See also the discussion at the end of V.C

The qualitative information collected indicates that the DODT helps developers to work in a more systematic way, formulate requirements more concisely, build a domain ontology and identify and correct requirement opacities and ambiguities. The quantitative information shows that building an ontology is an iterative process and that a considerable part – in our case 50% – of the original requirements may need to be re-written in the process.

Our experience is in line with that of other companies – see e.g. the experiences reported by Armagaud from AVL [21].

## VII. Further Work

The work on boilerplates and ontologies will be tested out in more case studies. In order to make any progress we need companies which have data that can be used as a baseline. We are especially interested in doing future research on the following topics:

- The number of inconsistent or opaque requirements that reaches the implementation stage.
- How much of the DODT effect stems from the ontology and how much which stems from the use of templates. We already know that Rolls Royce has had good experience with EARS, which is just a set of templates, while ABB claimed that most their benefits from the DODT stems from the requirements analysis based on the domain ontology.
- Will the use of DODT improve the communication and cooperation between the requirements engineers and the domain specialists

We have started to look for appropriate cases for such studies.

## References


[1] Standish Group: The Standish Group Chaos Report 2003

[2] S. Farfeleder, T Moser, A. Krall, T. Stålhane, H. Zojer, Ch. Panis: DODT: Increasing Requirements Formalism using Domain Ontologies for Improved Embedded Systems Development. Vortrag: 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2011), Cottbus, Germany

[3] Hull, E., K. Jackson, and J. Dick, Requirements Engineering. 2010, London: Springer Verlag

[4] Farfeleder, S., Moser, T., Andreas Krall A., Stålhane, T., Omoronyia, I., Zojer, H.: Ontology-Driven Guidance for Requirements Elicitation

[5] Fabbrini, F., Fusani, M., Gnesi, S. and Lami, G.: The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the use of an Automatic Tool. Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop

[6] Fabbrini, F., Fusani, M., Gnesi, S. and Lami, G.: Quality Evaluation of Software Requirements Specification. Proceedings of Software & Internet Quality Week 2000 Conference, San Francisco, USA

[7] Mavin, A., Wilkinson, P., Harwood, A. and Novak, M.: EARS (Easy Approach to Requirements Syntax). Proceedings of the 17th International requirements Engineering Conference

[8] Konrad, S. and Cheng, B.H.C.: Real-time Specification Pattern. Proceedings of the 27th international conference on Software Engineering

[9] Post, A., Menzel, I., and Podelski, A.: Applying Restricted Grammar on Automotive Requirements – Does It Work? A Case Study. REFSQ'11 Proceedings of the 17th international working conference on Requirements engineering: foundation for software quality

[10] Michael Grüninger, Christopher Menzel: The Process Specification Language (PSL) Theory and Applications. AI Magazine 24(3): 63-74 (2003)

[11] R. Bloem, A. Cimatti, K. Greimel, G. Hofferek, R. Könighofer, M: Roveri, V: Schuppan, and R: Seeber: RATSY - A New Requirements Analysis Tool with Synthesis. CAV 2010: 425-429

[12] Siegemund K., Thomas, E.J., Zhao, Y., and Assmann U.: Towards Ontology-driven Requirements Engineering. Proceedings of 7th International Workshop on Semantic Web Enabled Software Engineering (SWESE), October 2011.

[13] Falbo, R.A. and Nardi, J.C.: Evolving a Software Requirements Ontology. Em: XXXIV Conferencia Latino Americana de Informática - CLEI´2008

[14] Guizzardia, G., and Wagner, G.: A Unified Foundational Ontology and some Applications of it in Business Modeling. CAiSE Workshops (3) 2004

[15] Innab, N., Kayed, A., and Sajeev, A.S.M.: An Ontology for Software Requirements Modelling, 2012 IEEE International Conference on Information Science and Technology, Wuhan, Hubei, China; March 23 – 25, 2012

[16] Castaneda, V., et al.: The use of Ontologies in Requirements Engineering, Global Journal of Research in Engineering, vol. 10, issue 6, November 2010

[17] Kaiya, H. and Saeki, M.: Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. In Proc. of the 5th International Conference on Quality Software (QSIC), pp. 223-230, Melbourne, Australia. Sep. 19-21. 2005.

[18] ISO: Petroleum and natural gas industries – Design and operation of subsea production systems. ISO 13628-6:2002

[19] S. Farfeleder, T Moser, A. Krall, T. Stålhane, H. Zojer, Ch. Panis: DODT: Increasing Requirements Formalism using Domain Ontologies for Improved Embedded Systems Development. Vortrag: 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2011), Cottbus, Germany

[20] Rajan, A., Wahl, T.: CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems Springer, 25.March 2013

[21] Armnagaud, E. et al.: Integrated tool-chain for improving traceability during the development of automotive systems. Proceedings of ERTS$^2$ 2012

[22] Mavin, A. and Wilkinson, P.: "BIG EARS (The Return of "Easy Approach to Requirements Syntax"). 2010, 18th IEEE International Requirements Engineering Conference.