Establishing and Maintaining Traceability Between Large Aerospace Process Standards

Ove Armbrust,
Alexis Ocampo, Jürgen Münch
Fraunhofer Institute for Experimental Software Engineering (IESE)
Fraunhofer-Platz 1, 67663 Kaiserslautern Germany
{armbrust, ocampo, muench}@iese.fhg.de

Masafumi Katahira,
Yumi Koishi, Yuko Miyamoto
Japanese Aerospace Exploration Agency
2-1-1 Sengen, Tsukuba, Ibaraki 305-8505 Japan
{katahira.masafumi, koishi.yumi, miyamoto.yuko}@jaxa.jp

Abstract

The aerospace domain is known for its emphasis on product quality, since hardware or software failures may have potentially catastrophic consequences. Therefore, numerous standards govern space software development. In this paper, we present an approach for systematically establishing and maintaining traceability between software development standards. It augments the standards' word processing files with additional meta-information, thereby making them accessible and understandable for programs, so that a database can be used for advanced analyses. Additionally, we present experience collected during application of the approach at the European Space Agency (ESA) and the Japan Aerospace Exploration Agency (JAXA).

1. Introduction

The aerospace domain places great emphasis on the quality of both hardware and software, in order to satisfy rigorous security, safety, and reliability requirements. The application of suitable, well-described standards is one generally accepted way of achieving the desired quality. Such standards often come in the form of text documents processed in word processors such as Microsoft Word or OpenOffice.org Writer.

Since the subject covered is quite vast, a single standard is usually not sufficient to cover all topics. Therefore, a multitude of hierarchical, interrelated standards is a common condition. Examples of such standards include the ones published by the European Cooperation for Space Standardization (ECSS) [1] on behalf of the European Space Agency (ESA). The ECSS standards contain requirements for the development and management processes that any organization contracted by ESA must obey. Within ECSS, three series of standards govern the engineering processes (E series), the quality assurance processes (Q series), and the project management processes (P series). Other examples are the corresponding U.S. National Aeronautics and Space Administration (NASA) standards [2] or the ones prepared by the Japan Aerospace Exploration Agency (JAXA).

The hierarchical, heavily interrelated architecture of such standards poses some major challenges for process engineers. For example, tailoring individual standards becomes very complicated because of their relationships to other, possibly dependent standards. Evolving and maintaining the standards is equally challenging, because the dependencies of individual standards must be respected. In many cases, there must also be some sort of formal proof of compliance to a superior standard, e.g., when an organization-specific standard is developed. Finally, the application of advanced concepts such as process lines [3] [4] [5], which aims at transferring product lines concepts to the process world, also requires a large degree of control over the standards.

What is required for all these activities is detailed and high-quality traceability between entities of the individual standards. Establishing and maintaining this traceability is a labor-intensive, manual task, which is further complicated by the standard's representation as word processing documents. Therefore, the question we address in this paper is how traceability in a hierarchical, multi-standard environment can be established and maintained in a systematic, non-intrusive manner.

The paper is organized as follows. Section 2 provides a brief overview of existing approaches that seek an answer to this question. Section 3 presents our traceability approach, Section 4 outlines the benefits expected from its application. Section 5 presents two case studies in which the approach has been used, as well as their results with respect to the expected benefits. Finally, Section 6 draws some conclusions and gives an overview of future work.

2. Related work

A word processing approach similar to the one presented in this paper, but applied in a totally different context, is described in [6], where an XML editor was developed for editing legislative text. The XML format allowed processing such text and extracting a set of structured data (called metadata: act type, number, publication date, etc.) for different purposes.

Commercially available tools such as DOORS [7] support arbitrary traceability of software requirements. In our case, it was mandatory to keep the word processing files as working documents in order to allow editing and review by a variety of stakeholders.

In the software engineering community, traceability recovery from existing artifacts is increasingly important. Several approaches, for example based on Latent Semantic Indexing [8] or Ontologies [9], seek to assist humans with their tasks.

The need for an approach to engineering standardization in the aerospace domain is explained in [10]. An assessment model for assessing process compliance to standards has been developed and is known under the name of SPICE for Space (S4S) [11].

Traceability amongst different standards is especially important in software process line environments [3] [4] [5], where multiple variants are derived from one common core standard.

3. Database-supported traceability

For the rest of this paper, we assume that our mission is to establish traceability between a superior ("master") and a derived ("slave") standard. In order to achieve our objective of providing a systematic, non-intrusive way of establishing and maintaining such traceability, we propose the following steps:
(1) Define traceability rules.
(2) Augment the relevant documents with meta-information.
(3) Make pieces of information within the documents identifiable for programs through the use of styles.
(4) Establish traceability between the standards.
(5) Create and maintain a synchronized representation of the augmented documents in a database by processing their XML representation.
(6) Use the database representation to aggregate, analyze, and visualize the information gathered from the documents.

3.1. Define traceability rules

In order to control traceability between the standards, we propose defining traceability rules. A traceability rule describes allowed operations when deriving the slave from the master standard. For example, one such operation is to modify (or tailor) an entity. In this case, the traceability relationship between the slave and the master entity would be of the type "Tailored" (T). When analyzing traceability, this relationship indicates that the referenced entity is contained in both master and slave standards, but was modified in the latter.

Other relationships could be "Tailored Out" (TO) or "Unchanged" (U), indicating that the respective entity was removed or copied unchanged from the master to the slave standard. In the T and TO cases, some kind of explanation must often be given as to why this entity was modified or removed. We capture this information as a comment to the relationship.

3.2. Augment documents with meta-information

The next step ensures that the traceability information collected remains valid throughout the document's lifecycle. Since standards are highly structured documents, using sections as traced objects seems reasonable, but yields a major problem: Both the section number and its title may change. This means that these cannot be used to identify a section. We therefore propose adding a table with some additional meta-information to each section, which enables us to identify sections at any time.

Table 1. Example of a meta-information table

Meta-information		
ID	*invariant ID of the respective section*	
Change log	*description of changes*	
Reviewer comments	*reviewer feedback*	
Traceability		
Master ID	Status	Comment
master section ID	*{U, T, TO}*	*comment on status*

Table 1 shows an example of such a meta-information table. The "ID" is a unique, invariant ID that identifies the section. The "Change log" allows editors to describe the changes they made to the section with respect to its previous version – thus providing a very detailed record of all modifications to the document. "Reviewer comments" will be used by reviewers to give feedback on the respective section. The "Traceability" part of the table holds the information on which parts of the master standard the respective section is related to, the traceability status, and any comment explaining the status, e.g., what was changed with respect to the master standard and why (in the case of a "T" status).

3.3. Make information program-identifiable

Once the sections can be identified, we need to be able to identify certain types of information within the document. For humans, it is obvious that a piece of text such as "Expected output: functional requirements" denotes a work product that is expected to be produced at this point. For a program, however, this is not the case. Therefore, we propose providing additional help to a program so that it can identify different types of information through the use of styles.

In modern word processors, styles are used to define the formatting of a piece of text. For example, in this paper, the style "Heading 1" is defined as 12-point Times bold font. This style is applied to all level-1 headings, effectively telling the word processor to display all such headings in bold, 12-point Times. While this makes it simple to give documents a consistent look, it also provides a very useful side effect: The word processor memorizes that a certain piece of text is assigned the "Heading 1" style.

This, in turn, is extremely helpful when a program is supposed to analyze the text and extract information from it. For example, it now becomes possible (and actually simple) to extract all level-1 headings from this document – by simply extracting all text that is assigned the "Heading 1" style. In the case of our standards, we assigned all inputs for an activity a style called "Input" and all outputs a style called "Output", enabling us to extract this information from the document later.

3.4. Establish traceability

Now that the documents have been prepared, we are able to establish, for every section of the slave standard, traceability to the master standard. Even though this is a human-centered task, appropriate tool support can help speed up the process [12]. Nevertheless, it requires understanding both master and slave sections in order to evaluate whether (a) they have any relationship at all and (b) if so, what kind or relationship (U, T, TO) this is. Our approach alleviates this problem partially by enabling editors to record the traceability within the respective section (see Table 1, "Traceability" part) instead of in a different document.

3.5. Create and maintain synchronized database representation of documents

Now all preparatory work is complete. The next step is to create a database representation of the documents. Saving word processor files in XML format makes them accessible to external tools such as XML parsers. We used one such parser to extract the relevant information. Essentially, we instructed the parser like this: "For every activity description (i.e., section), extract its ID, title, heading number, change log, reviewer comments, and traceability information. Additionally, extract all inputs and outputs. Write all the information to the database." All this was accomplished with a few hundred lines of Python code [13]. What we got in return was a 1:1 representation of the standard in our database – ready to be analyzed. Every time the standard is edited, its database representation is updated, so the two are always synchronized.

3.6. Aggregate, analyze, visualize

Now that we have the essence of our standard in a program-interpretable form, what can we do with this? Aggregate data, for example. Questions like "Which outputs are created during the requirements elicitation phase?" are quickly answered by a single database query. It is also possible to get a quick overview of the traceability status (which parts of the standard are highly tailored, which ones are tailored out, …). We can also check for inconsistencies (e.g., outputs that are required, but never produced anywhere) or create visual representations of the standard (see Figure 1).

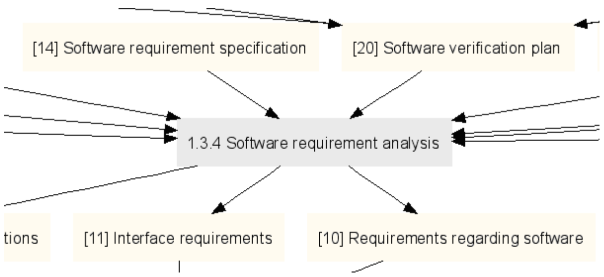
[14] Software requirement specification
[20] Software verification plan
1.3.4 Software requirement analysis
tions
[11] Interface requirements
[10] Requirements regarding software

Figure 1. Standard visualization example

4. Expected benefits

We expect a number of benefits from the approach taken. First, and most important, we expect to continue using standard word processing software such as Microsoft Word or OpenOffice.org Writer. This is especially important, since the editors of standards often use these tools. Forcing them to use other, specialized tools would probably lead to low acceptance of any such solution and make other activities such as reviews or information exchange more complicated.

Second, we expect significant advantages concerning internal (i.e., intra-standard) consistency. For example, if a program can identify different kinds of artifacts and their relationships, it can also check whether some designated illegal conditions such as orphaned outputs exist. Therefore, we expect that we can prove the absence of some of these illegal conditions through automated checks.

Third, we expect similar benefits concerning external (i.e., inter-standard) consistency. For example, we expect to be able to guarantee that every entity in the master standard is considered, and that for every slave entity, it is clearly defined which master entity it corresponds to. Thus, we expect to be able to guarantee that no entity is left behind.

Finally, the database approach promises a number of further benefits, such as transparency on the degree of tailoring (how many entities were removed/added/ modified), on the implications of changes to one standard to other, derived standards through the traceability information, and on standard quality through the automated consistency checks and, hence, better directed reviews by humans.

5. Case studies

We applied the approach described in two different environments. The first application was at ESOC, the European Space Agency's Ground Segment. Traceability was established between ESA's software standards (ECSS E-40b, Q-80 and a number of M standards) and ESOC's tailoring of these standards for Ground Segment work. The ECSS standards were available in PDF form. ECSS states requirements that processes in ESA projects must comply with, in our case about 1,600 of them.

ESOC's tailoring of ECSS is called SETG (Tailoring of ECSS Software Engineering Standards for Ground Segments in ESA) [14] and consists of two OpenOffice.org Writer/ Microsoft Word files describing the activities and artifacts (SETG A and B), one file describing the document templates (SETG C), and one file containing all traceability information (SETG D). Traceability was established between ECSS requirements and SETC A and B, which together have about 110 content pages. Within ECSS, every requirement was traced, together with expected outputs and their respective reviews. Traceability targets within SETG were activities (e.g., "Determine Interface Requirements") and outputs. Thus, for every requirement and expected output within ECSS, ESOC is able to name the respective activity and output that realizes the ECSS demands.

The second application was at the Japanese Aerospace Exploration Agency (JAXA). JAXA maintains a hierarchical standards architecture. From level-1 standards such as ISO/IEC 12207 [15] or ECSS, the generic JAXA level-2 software development standard is derived. The level-2 standard, in turn, is used as a basis for a number of level-3 standards, which are domain-specific, e.g., for satellites, launch vehicle, ground segment. These domain-specific standards are further tailored for specific projects (level 4). So far, we have been working on level-2 and level-3, with expansion to level-1 and level-4 standards planned for the future.

All JAXA level-2 and level-3 standards were available as Microsoft Word documents. Traceability was done per activity and per input and output in both the level-2 and the level-3 standards. Since this project has not been completed yet, all experience is preliminary, but points out important trends nonetheless.

Considering our expected benefits, we can fully confirm our first expectation ("continue using standard word processing software"). For the editors of the standards, the additional table per section for the meta data and the modified form of writing down inputs and outputs were only minor modifications to their mode of operation. Especially recording the traceability information directly within the respective section / next to the respective artifact was by far more user-friendly than any external spreadsheets or similar solutions. Not using any special tools allowed the extended workflow (international reviews, information exchange, etc.) to function as usual.

Our second expectation ("significant advantages concerning intra-standard consistency") was also fully met. For example, automated checks revealed a number of inconsistencies within the product flow, such as artifacts being used, but never produced before, or vice versa. Detecting *all* such inconsistencies in a standard with dozens of activities on multiple hierarchy levels and hundreds of inputs and outputs *manually* consumes extreme amounts of effort and most probably still misses some inconsistencies. An automated check took only seconds and detected all such inconsistencies for sure – thereby relieving human reviewers of this boring and tedious task.

Finally, our third expectation ("similar benefits concerning inter-standard consistency") did not disappoint us, either. Through database analyses, we could, in fact, prove that every entity of the master standard was accounted for (i.e., had at least one incoming link of type T or TO or U). Similarly, we could prove for every slave entity that its traceability to the master standard was established (i.e., it had at least one outgoing link of type T or TO or U). During approval of the SETG standards by ESOC's Engineering Standardization Board (ESB), the minute traceability reports for all ECSS requirements simplified the approval process significantly. The JAXA traceability project is not yet at such a point, but we expect it to yield similar benefits.

We encountered a number of additional benefits, which we will only mention here due to space restrictions. During the JAXA project, we created a very simple visualization of the intra- and inter-standard relationships (see Figure 1), which helped engineers understand the complexity of the standards. When one standard was changed, we could immediately name the sections of other standards that must also be reviewed because of this change. The guaranteed absence of certain types of inconsistencies and, hence, the ability to focus reviews on aspects that really require human understanding led to an increased standard quality – although it is difficult to quantify. Also, the integration of comment fields for standard reviewers into each activity description of the standard simplified the collection and aggregation of the review comments.

6. Conclusions and future work

We have presented a systematic, non-intrusive approach for establishing and maintaining traceability between software standards that are represented as word processing documents. By augmenting these documents with additional information and by making individual pieces of information identifiable for programs using styles, we were able to create a database representation of the standard, which we then used for advanced analyses and visualization.

The approach fully satisfied our expectations, and provided even more benefits than initially anticipated. The document-based traceability records were especially helpful and user-friendly. We have only just begun to explore the powerful analysis capabilities provided by the database representation of the standards. Automated consistency checking relieved human reviewers of some boring and error-prone tasks, so that they could concentrate their attention on other tasks that required human understanding. The case studies indicated that the approach is feasible for large standards, such as those for the aerospace domain. From our experience, we expect the approach to work as well in other domains with adequately documented process standards.

In the future, we will continue exploring the possibilities provided by the database representation of the standards, and further improve the support for standard editors through reports on more issues.

7. References

[1] European Cooperation for Space Standardization, http://www.ecss.nl/, last visited 2009-01-21.

[2] NASA Technical Standards Program, http://standards.nasa.gov/default.taf, last visited 2009-01-21.

[3] H.D. Rombach, "Integrated Software Process and Product Lines", Lecture Notes in Computer Science 3840/2006, Springer Berlin / Heidelberg, 2006.

[4] O. Armbrust, A. Ocampo, "Software Process Lines and Standard Traceability Analysis", 7th Workshop of Critical Software, January 14-15, 2009, Tokyo, Japan.

[5] O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, A. Ocampo, "Scoping Software Process Models - Initial Concepts and Experience from Defining Space Standards", International Conference on Software Process (ICSP), May 10-11, 2008, Leipzig, Germany.

[6] M. Palmirani, R. Brighi, "Metadata for the legal domain", 14th International Workshop on Database and Expert Systems Applications, Springer Verlag, Berlin Heidelberg New York, 2003, pp. 553- 558.

[7] Telelogic DOORS, http://www.telelogic.com/, last visited 2009-01-10.

[8] A. de Lucia, F. Fasano, R. Olivetto, G. Tortora, "Enhancing an artefact management system with traceability recovery features", 20th IEEE International Conference on Software Maintenance (ICSM), Chicago, IL, USA, 2004.

[9] Y. Zhang, R. Witte, J. Rilling, V. Haarslev, "An Ontology-based Approach for the Recovery of Traceability Links", 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM), Genoa, Italy, 2006.

[10] J.-L. Cendral, M. Merri, R. Choda, "Engineering Standardization", European Space Agency Bulletin 122, 2005.

[11] A. Cass, C. Völcker, L. Winzer, J.M. Carranza, A. Dorling, "SPICE for SPACE: A Process Assessment and Improvement Method for Space Software Development", European Space Agency Bulletin 107, 2001.

[12] A. de Lucia, R. Oliveto, G. Tortora, "Assessing IR-based traceability recovery tools through controlled experiments", Empirical Software Engineering 14 (1), Springer Netherlands, 2009, pp. 57-92.

[13] M. Lutz, "Programming Python", 2nd Edition, O'Reilly & Associates, Sebastopol, California, USA, 2001.

[14] BSSC Guides and Reports, http://www.esa.int/TEC/Software_engineering_and_standardisation/TECT5CUXBQE_2.html, last visited 2009-01-21.

[15] International Organization for Standardization, ISO/IEC 12207:1995, Geneva, Switzerland, 1995.