Tracing Process Model Evolution: A Semi-Formal Process Modeling Approach

Ove Armbrust, Alexis Ocampo, Martín Soto

Fraunhofer Institute for Experimental Software Engineering
Fraunhofer-Platz 1
67663 Kaiserslautern
Germany
{armbrust, ocampo, soto}@iese.fraunhofer.de

This paper describes our experiences during the evolution of a text-based process description within the aerospace domain. We describe our tool-assisted way of editing and reviewing the process description and the way we traced elements of the edited process description to a superior standard in order to prove compliance. To achieve this, we created a persistent connection of standard word processor documents to a model of the documents in a relational database, which allowed us to keep arbitrary meta information as well as to automate advanced consistency checks and the collection of review comments. The approach worked well within our environment and was flexible enough to incorporate additional requirements during the project.

1 Introduction

In the aerospace domain, all parts (including software) must adhere to rigorous quality standards due to high requirements on security, safety, and reliability. Describing the software development process in detail helps to achieve the required quality. Since large aerospace projects are not carried out by one single country, Europe-wide standards define a framework for the process definitions that govern individual projects.

One such Europe-wide framework is provided by the European Cooperation for Space Standardization (ECSS) [1]. The ECSS is an initiative established to develop a coherent, single set of easy-to-use standards for all European space activities, covering all areas of space activities, including engineering, quality assurance, and project management. Since the ECSS standards are *standards for standards*, they are basically a set of requirements demanding certain things to be done, but not determining how. This way, a common framework exists that enables interchange among developers and organizations while preserving the necessary flexibility.

2 Our Mission

The requirement-based framework can make it rather difficult to directly use the ECSS standards. This is intended; in fact, local agencies of the European Space

Agency (ESA) are supposed to develop and use their specific tailoring(s) of the ECSS standards. The tailoring can be done project-specific (i.e., a separate tailoring for every project) or organization-specific (i.e., one tailoring per organization, to be used for all their projects). Our partner within ESA chose the organization-specific tailoring approach. The applicable implementation of their ECSS tailoring was the Software Engineering and Management Guide (SEMG) [2], which was used for all their major projects.

After some years of experience with the ECSS standards, they were revised by ESA, and a new version was published. This also meant updates to the SEMG, in order to be compliant to the revised ECSS standard. Our goals in the project were as follows:

- Tailor the relevant parts of the ECSS (comprising several hundred requirements) to our partner's needs and apply this tailoring in an update of their implementation of the standard, the SEMG.
- Prove the compliance of the SEMG to the ECSS by providing traceability and tailoring information on every ECSS requirement to its implementation within SEMG.
- Improve the ease of use of the SEMG by improving (1) internal consistency, i.e., one part of the document does not contradict another, (2) external consistency, i.e., the document at hand does not contradict other documents, links to external sources are correct, and (3) conciseness, i.e., indexed tables of content allow people to find important things quickly, different concepts are explained and marked clearly, and the document is not larger than necessary.
- Because of very different stakeholders and stringent quality requirements, detailed change logs were required on a per-section basis.

This paper describes our approach to the SEMG update, as well as our experiences. The paper is structured as follows. Section 3 introduces some related work. Section 4 describes the analysis we performed initially. Sections 5 and 6 describe our approach and the editing of the SEMG documents, including reviews. Section 7 presents our conclusions. Possible future work is discussed in Section 8.

3 Related Work

The field of software process modeling has become an established field within the software engineering community. Our particular approach descends from the methodological and practical work done at the Fraunhofer Institute for Experimental Software Engineering (IESE) on the Spearmint^{R} process modeling tool [5], which, in turn, is based on the concepts of the MVP-L [6] modeling language.

A word processing approach similar to the one presented in this paper, but applied in a totally different context, is described by [7], 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 [8] 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 for editing and review by a variety of stakeholders.

The need for an approach to engineering standardization in the aerospace domain is explained in [9]. An assessment model for assessing process compliance to standards has been developed and is known under the name of SPICE for Space (S4S) [10].

4 Compliance and Ease-of-Use Analysis

Our first task was to analyze the SEMG standard for two things: First, compliance to the updated ECSS standards, and second, flaws that reduce the ease of use. Both analyses were executed manually.

The compliance analysis consisted primarily of a concentrated study of the SEMG and a consecutive check for every ECSS requirement of where in the SEMG the respective requirement is implemented. The SEMG was considered compliant to ECSS if and only if every applicable ECSS requirement is either fully implemented, or "tailored", for specific, listed reasons. The tools we used during this phase were simply printouts of the documents and pencils. We made notes on the requirement compliance directly in the ECSS document, referring to the SEMG sections.

Our analysis showed that the SEMG was only partially compliant to the new ECSS software standard, and had to be updated accordingly. The analysis process showed that compliance was hard to determine in the first place and even harder to prove, because of the very different structure of the two standards. Within the SEMG, many relationships were noted only implicitly, making it hard to track and prove them. It also made it hard to understand the standard.

The ease-of-use analysis was done by analyzing the SEMG documents and by means of structured interviews with SEMG users. From hearsay, we already had the impression that the SEMG users were particularly unhappy with the number of documents they had to provide during the software lifecycle. Thus, we concentrated on these documents and their contents in the interviews. In total, we interviewed nine people from different sections. All were in leading positions (e.g., section heads or heads of development). We collected people's overall impression of the SEMG and details about the production of documentation. Both types of information were used to prioritize SEMG parts for enhancement. The most predominant wish was for output document simplification and clarification. Furthermore, the SEMG structure did not reflect actual process execution any more and had to be adjusted accordingly.

5 Our Approach

The initial analysis showed us the necessity of establishing a reliable traceability mechanism to capture, display, and maintain the compliance information between the ECSS standards and the SEMG. Additionally, since our partner required us to provide a way to review any changes made, we needed to keep a log describing and justifying any actions performed on the document contents. This combination of external and internal consistency demands, as well as the previously mentioned usability issues,

increased the complexity of evolving the SEMG standard, and motivated the need for a systematic strategy supported by an appropriate technological infrastructure. Therefore, an iterative cycle for editing and reviewing the SEMG was established. Two iterations were performed in order to incrementally edit and improve the SEMG standard to its final version. During the editing of the SEMG, process engineers concentrated on improving compliance to the ECSS standards, and its usability. During the review phase, changes to the SEMG were accepted or rejected, and comments and new suggestions were provided by the reviewers.

5.1 The Formal Model

Due to the complexity of the traceability information, an automated solution was needed in order to reliably store and maintain the traceability relationships between elements of the involved standards. The first step in that direction was to define a formal model containing the logical elements needed to store traceability and change information. This model took the form of a traditional entity-relationship diagram that we later used as the basis for implementing a simple software system that relied on a relational database.

Fig. 1 presents an excerpt of the entity-relationship diagram. It shows one entity representing the SEMG sections and another one representing the ECSS requirements. These entities are connected by a traceability relationship, corresponding to the fact that a given SEMG section implements, or contributes to implementing, a particular ECSS requirement. The relationship also contains information about the degree to which the referenced requirement is implemented by the referenced section. Entities also contain additional information, for example, an *SEMG Section* entity features a *Change log* field for storing information on changes performed to that section.

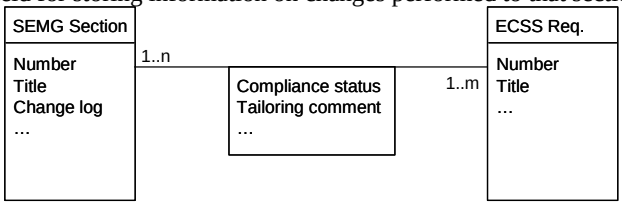

Fig. 1. The initial model

5.2 The Technical Infrastructure

Based on the previously described model, we implemented a database schema and filled the corresponding tables with the basic information. This was done by importing and filtering the SEMG and ECSS section numbers and titles automatically from the electronic documents, which saved us the tedious work of entering this data by hand, thereby allowing us to concentrate on the traceability relationships.

Although the database solution proved to be adequate for holding the traceability information, it was soon clear that once we started modifying the SEMG (potentially moving, renaming, adding or deleting sections), keeping the database synchronized

with the document was going to be difficult. The main problem was that at first, we were identifying document sections using only their numbers and titles, but both of those values could change for a given section during editing. This implied that relatively simple document changes, such as deleting a section, would result in potentially very complex updates to the database.

The way to handle this problem was to assign a unique identifier to every section, which was to remain unchanged during document lifetime. Due to the fact that such identifiers had to be kept attached to their respective sections during the whole editing process, we decided to store them as part of the document content. In this way, someone editing the document would only have to take care of keeping the identifiers together with the corresponding text, which is not a difficult task to accomplish with a standard text editor.

At the same time, we noticed that this approach was also useful for other information items relevant to a section. This led us to inserting a number of fields at the beginning of each document section, contained in a standard word processor table. Fig. 2 shows an example of such a table, which we call a *meta-information table*. The table contains not only the section's unique identifier (field *invariant id*) but also a change log and a list of traceability relationships. The fields for storing the traceability relationships (*ECSS Coverage* section) correspond to the respective fields in the database, with the *Comments* column capturing the rationale for the coverage. A field for collecting reviewer comments was added later (see below for more details).

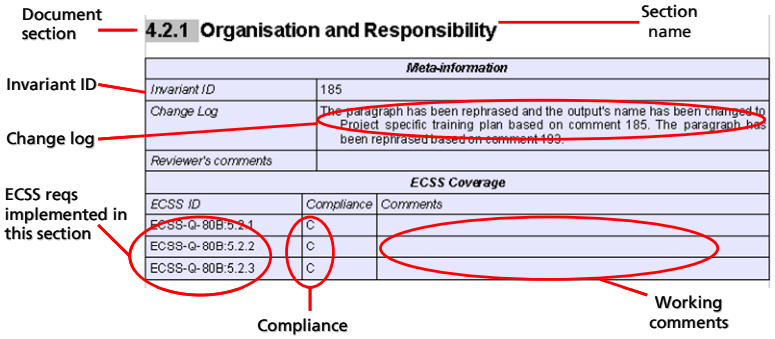

Fig. 2. The meta-information table

5.3 The Automatic Document-Database Update Mechanism

The information stored in the meta-information tables in the document is the same as the information in the database and thus redundant. However, both representations can be useful depending on the task at hand. For example, the document representation is especially useful while manually editing the information, whereas the database representation is much more practical for performing automated analyses.

For this reason, the next logical step was to devise a mechanism to automatically and reliably keep both representations synchronized. Our solution relied on the fact that modern word processing programs increasingly support the Extensible Markup Language (XML) as a document format [3]. As an open format, XML can be proc-

essed using a variety of widely available tools, including high-level libraries that can be invoked from most modern programming languages.

Using the interpreted, object-oriented Python programming language [4], we developed a parser that is able to navigate through the XML versions of the SEMG documents, identify the section headings and meta-information tables, and move information to and from the database as necessary. This functionality allowed us to update the database automatically after a set of changes, and to check the data for consistency before doing any further editing. Also, whenever it was necessary to update information directly in the database, our system was able to update the meta-information in the documents accordingly.

6 Document Editing

The actual editing process used both the ECSS and SEMG standards as inputs. Basically, the strategy was to repeatedly rely on the information in the database to identify ECSS requirements that were not covered (or were only covered partially) and proceed to extend or modify the document to ensure the coverage. Since such changes often implied adding new sections or modifying existing sections to cover additional requirements, the traceability relationships had to be updated accordingly. This could always be done directly in the document and transferred to the database automatically.

As already mentioned, while editing, we had to fulfill two central requirements: to make the SEMG compliant to the ECSS standards and to increase its ease of use. One fundamental step towards improving the ease of use was to restructure the document sections by organizing them into *Purpose*, *Description*, and *Outputs* subsections, which were marked explicitly using word processor formatting styles. This was a valuable decision because we were later able to rely on this mark-up to perform further automatic processing of the document.

6.1 Model Extension

Late in the editing process, and after an informal review of the work in progress, the stakeholders arrived at the conclusion that the terminology used in the SEMG to refer to the process outputs was too abstract in comparison to that used in the ECSS standards, a fact that could lead to confusion. For that reason, we had to extend our formal model to include a new *ECSS Output* entity, as well as a number of other entities and relations related to it, such as *Review* and *Document Template*. The necessary data was imported into the database directly from the electronic ECSS documents, and later used to automatically generate the relevant *Outputs* subsections in the SEMG.

6.2 Consistency Checking

Simultaneously with the editing process, we developed a number of automated mechanisms to check for different types of consistency as the SEMG evolved. One mechanism consisted of defining constraints in the database management system that

checked for basic data restrictions, like, for example, warranting that all document section identifiers mentioned in the database actually existed as such. Another one was realized through Python programs that combined SQL database queries with procedural programming in order to check for more complex consistency conditions or other types of potential errors. For example, we wrote programs for finding outputs not assigned to a review or not assigned to a section in a document template.

6.3 Document Review

As mentioned before, reviews were performed by the stakeholders in order to approve or reject changes, and provide comments and/or suggestions. In order to accomplish this, we included a *Review Comments* field (see Fig. 2) in the meta-information tables (and extended the formal model accordingly) and sent each one of the reviewers an electronic copy of the document where they could directly type in their comments. Once the review was finished, and using the already established infrastructure, all comments were automatically collected into summary tables including fields for action tracking. This made it possible to easily create review reports including all remarks, answers, decisions, and changes.

For the final review the stakeholders not only received the final edition of the SEMG, but also traceability reports, detailed change logs on a per-section basis, and tailoring information. After the review, we generated a version of the document intended for the process users, which did not contain any meta-information used during the review. Our infrastructure played a key role in making this possible.

7 Discussion

Despite certain difficulties arising from applying our approach for the first time to a practical, real world project, we consider our results quite satisfactory. The approach not only proved to be suitable for the set of problems at hand, but showed a clear potential for being applicable to future similar problems. The following subsections describe the aspects that, in our opinion, are most relevant for future use.

7.1 Semi-Formal Process Modeling and Degree of Formalization

Our approach to process modeling can be called semi-formal. On the one hand, the main process description is a formal model, based on formally defined process elements and relationships (mainly the database schema in this experience). On the other hand, many aspects of the elements in the model are described using (informal) natural language, to the extent necessary to make the model usable. In actuality, the natural language descriptions represent the bulk of the modeling effort, and are fundamental to understanding and using the model. This approach can be contrasted with both formal approaches, which attempt to describe every single aspect of the process using a formal notation and associated semantics, and informal approaches, which rely on text descriptions that, although structured, do not follow any formalized schema.

A central aspect in semi-formal modeling is that a process engineer can decide the extent to which a process description is formalized. This is, in our opinion, one of the main advantages of the approach. By tailoring the degree of formalization, it is possible to strike a balance between the simplicity of informal modeling, and the low ambiguity and analysis possibilities offered by formal modeling.

Additionally, as the present experience shows, an informal process description can be gradually formalized by defining a schema that corresponds to its basic structure, and extending it in subsequent steps as determined by the needs of a particular project. This constitutes an effective option for improving the quality of an existing description without incurring the costs and higher complexity of a complete formalization.

7.2 Cost Effectiveness

Complexity, and its ensuing high costs, are probably the most serious risks in a formal process modeling effort. Our experience shows that by selecting an adequate degree of formalization, cost and complexity can be kept under control, without sacrificing most of the benefits provided by formalization.

In this project, the cost of the formalization effort (setting up a formal model in the form of a database schema, populating the resulting database), was, in our judgment, clearly lower than that of performing the analysis and consistency checks completely by hand. The consistency checks performed in the project were not only extensive, covering several hundred document pages, but quite complex in some cases. Doing this work manually would have been very labor intensive and error prone.

By partially formalizing the involved process description first, and then performing the analysis automatically (or manually with automated assistance) on the resulting model, a good deal of tedious work was avoided, allowing us to dedicate our time to more productive tasks that actually required human attention. Additionally, in our opinion, this avoidance of tedious manual work also led to more consistent and reliable results.

Further efficiency and reliability were gained by keeping the meta-information related to a particular process element as close to the main element description as possible. This prevents distractions associated with switching work contexts, and makes it much easier to perform manual reviews and verifications.

7.3 Transparency

As already explained, transparency played a very important role in this project. Particularly, keeping track of all changes made to the documents, not only in terms of what was changed, but also in terms of the reasons behind each change was absolutely necessary. Achieving this was only possible by providing automated support for change tracking. Particularly, attaching the traceability and change information directly to each of the document sections, and collecting it automatically from there, made the whole process not only more comfortable, but much more reliable.

The attached logs strongly reduce the burden of maintaining a large, completely separate change log that must be manually cross-referenced to the main document (a highly error-prone process). The final result was an extensive change database, providing a high degree of accountability for the performed process engineering work.

An additional, very valuable step in terms of guaranteeing the transparency of the process was the automated support for the external review rounds, allowing the reviewers to directly type their comments into the documents. This made it straightforward to associate the comments reliably to the affected document sections, and to properly keep track of them while addressing the discussed issues. This, in turn, contributed greatly to providing the reviewers with the assurance that their concerns were properly taken into account.

7.4 Flexibility

The need for flexibility was another central aspect of this project. The complexity of the decision process behind the involved standards, as well as the large number of stakeholders, led to a permanent flow of new and changing requirements that had to be addressed as they arose.

Both the general process modeling approach and the choice of technology proved to be up to the task. Having the possibility of easily moving information between the documents and the database made it relatively easy to extend and modify the formal schema as necessary without seriously disrupting our work flow.

7.5 Technology Choice

We think that our technology choice played a central role for the success of this project. First of all, being able to produce easily processable XML documents directly from a standard word processing application was instrumental to many of the tasks we performed in the project. On the one hand, using a standard word processor (as opposed to a specialized process modeling or requirements management application) not only made our work easier and more comfortable but allowed us to interact with other project stakeholders in a straightforward way. On the other hand, having such documents readily available in XML form provided us with a wide choice of technologies to analyze and process the data.

The use of a high-level, interpreted programming language (Python) in combination with a relational database system also showed to be an appropriate choice. The high-level nature of the language clearly reduced the amount of programming and debugging time necessary, thus making it possible for us to extend the infrastructure with reasonable effort.

8 Future Work

Our experience in this project pointed to a number of aspects that remain open for future research. First of all, our approach to identify the elements that had to be for-

malized was completely ad-hoc: we started with a minimal set of required information and expanded it as necessary. Although our technical infrastructure was able to handle this, it would be beneficial to better know in advance which elements to formalize. Furthermore, although the use of a relational schema was sufficient for this project, we are aware of the fact that other notations may be more adequate in the general case. We are currently investigating other formalization possibilities in order to provide a more systematic way of using our approach.

Additionally, although the effort described here concentrated on the aerospace domain, we believe that our approach is suitable for a wider variety of uses. We are currently trying to extend it to the domain of service grids and are seeking validation opportunities in other fields.

Acknowledgements. We would like to thank Michael Jones and Mariella Spada from ESA Space Operations Center (ESOC) and Jürgen Münch from Fraunhofer IESE for their support and their valuable comments. Additionally, we would like to thank Sonnhild Namingha from Fraunhofer IESE for reviewing the first version of this paper. We also appreciate the valuable comments of the anonymous reviewers.

References

1 European Cooperation for Space Standardization (ECSS), standards available at http://www.ecss.nl. Last checked 2005-08-18
2 BSSC documents. http://www.estec.esa.nl/wmwww/EME/Bssc/BSSCdocuments.htm. Last checked 2005-08-18
3 Merz, D.: XML for Word Processors. Open Source Embraces XML as Native Document Format. IBM Developer Works: 25 February 2004. Available at: http://www-128.ibm.com/developerworks/library/x-matters33/
4 Lutz, M.: Programming Python (2nd Edition). O'Reilly & Associates, Sebastopol, California (2001)
5 Becker-Kornstaedt, U., Hamann, D., Kempkens, R., Rösch, P., Verlage, M., Webby, R., Zettel, J.: Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance. Lecture Notes in Computer Science, Vol. 1626. Springer-Verlag, Berlin Heidelberg New York (1999) 119-133
6 Bröckers, A., Lott, C.M., Rombach, H.D., Verlage, M.: MVP-L Language Report Version 2. Internal Report Nr. 265/95. Department of Computer Science, University of Kaiserslautern, Germany (1995)
7 Palmirani, M.; Brighi, R.: Metadata for the legal domain. In Proceedings of the 14th International Workshop on Database and Expert Systems Applications, Springer Verlag, Berlin Heidelberg New York (2003) 553- 558.
8 Telelogic DOORS, http://www.telelogic.com/, last checked 2005-10-06.
9 Cendral, J-L., Merri, M., Choda, R.: Engineering Standardization. European Space Agency (ESA) Bulletin 122 (2005)
10 Cass, A., Völcker, C., Winzer, L., Carranza, J.M., Dorling, A.: SPiCE for SPACE: A Process Assessment and Improvement Method for Space Software Development. European Space Agency (ESA) Bulletin 107 (2001)