Model-Based Requirements Authoring

Creating Explicit Specifications with OPM

Alex Blekhman
Technion, Israel Institute of Technology, Haifa, Israel
blekhman@tx.technion.ac.il

Dov Dori
Technion, Israel Institute of Technology, Haifa, Israel
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
dori@ie.technion.ac.il

Abstract. We present a general process, supported by Object-Process Methodology (OPM) for authoring model-based requirements specifications. Requirements specifications are first conceptually analyzed and an OPM model of the technical content is created. Then, constrained natural text is derived and adjusted based on the model. This approach is guided by adopting general model-based systems engineering principles to requirements authoring and can be extended and applied broadly for authoring various kinds of technical documents.

I. INTRODUCTION

A central part of a systems engineer's work is creating technical specifications. A typical requirements document authoring scenario seems to be simple at first glance, as all that is needed is to create a clear presentation of the function and characteristics required from some the system, product, or service in demand. Describing the function and the main components soon turns out to be complicated, though, as it gets to finer details such as technical features, operating modes, and interfaces, where paying attention to specific fine points while maintaining a clear system vision and avoiding contradictions becomes a real challenge.

As the scenario unfolds, following remarks from peers, superiors, or customers, changes are inevitably made to the specification at various levels, spanning from configuration and high level components to measurement units, weights, etc. These changes have their effect on the acceptance and test plan, adding some new demands and removing some old statements. Sometimes the mission can be temporarily abandoned and then resumed, or the specification can be reviewed by a colleague that has her own personal style and vision. These unavoidable changes to the text leave little chance for the document to remain consistent and precise.

Does this scenario seem all too familiar? ISO (International Standards Organization) has encountered this problem from another perspective0 (Dori et al, 2010). ISO standards, as well as those of other standards organizations, are often criticized as being difficult to use for a variety of reasons, including inter- and intra-standard inconsistency, low accessibility, poor traceability, and ambiguity. Indeed, given the variety of authors and relationships to other domain standards on top of the usual hurdles renders managing the quality of a technical document such as an ISO standard a daunting task.

A major root cause for this state of affairs is the lack of an underlying analytical process that would accompany the creation and maintenance of technical documents, such as requirements, directives, and standards. This process should provide means for technical document verification and validation, evaluating its scope and implications, and relaying and comparing it to other relevant technical materials. Nevertheless, in order to provide means of communication amongst different, possibly non-technical stakeholders, the end result of the process should have text as the main communication modality.

We have started to explore the issue of text-based document management with International Standard IEC/FDIC 62264 (Enterprise-control system integration) as a test case0 (Johnsson, 2003, Blekhman et al, 2010). We soon found that converting information to model-based structured form helps pinpoint and resolve ambiguities and refine the author's intent. Furthermore, the extent of inconsistencies we were able to detect while modeling the detailed textual information encouraged us to develop a synthesis procedure for authoring technical documents.

The major objective of this approach is to convey text and figures in a consistent form, so that the rectified text stemming from the model is composed of simple, light, unambiguous sentences. Not less importantly, this text is fully aligned with its underlying Object-Process Methodology (OPM) model.

An early application of this model-based approach to authoring technical documents is the Model-Based Requirement Authoring (MBRA) process, which is the focus of this research. The rest of the paper is organized as follows: Section 2 surveys related work in the fields of requirements engineering and specification authoring. Section 3 presents the general approach that guides our view of requirement authoring and documents authoring in a broader sense. Section 4 contains a review of Object-Process Methodology (OPM), which is then presented as an underlying structural formalism that accompanies MBRA. Section 5 contains an example that illustrates the stages and the benefits of our method in a self-reflective manner. Section 6 concludes with the merits of the proposed methodology and suggests directions for its future utilization and extension.

II. RELATED WORK

Requirements specification is often guided by manuals and templates that mostly specify the structure of the document, without reference to the desired contents, except for syntactic and deontic rules, e.g., using 'shall' instead of 'may', etc. Most of these guides prescribe only generic principles for creating good requirement documents. IEEE Std. 830-1998**Error! Reference source not found.** (IEEE 2008), for example, which relates to Software Requirement Specification – SRS, states that it should be "… correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable and traceable" (section 4.3, p. 4). While these principles appear to be intuitive and concise, summarizing years of common experience, it is difficult to translate them into practical implementation, as the requirements engineer is given very little, if any, concrete guidance, tool, or procedure.

Much effort is made in order to bridge the gap between requirements and systems engineering in general and systems architecture in particular (Grünbacher et al, 2001). Yet, model-based representation of technical requirements is not mature enough to cope with the problems described above. Existing approaches are not yet ready for prime-time application, as they are mostly focused on narrow fields and purposes.

Most requirement modeling techniques are intended to be used in an analysis phase for constructing an analytical model once the requirement specification is given. One effort of this kind is System Model Acquisition from Requirement Text (SMART, Dori et al 2004). Enterprise Architect's requirement management and modeling tool (http://www.sparxsystems.com.au/downloads/whitepapers/Requirements_Management_in_Enterprise_Architect.pdf), which is a commercial tool, also employs modeling during the requirements analysis phase rather than during synthesis.

Representation tools, requirement methods and languages can be categorized into object-oriented0 (Kasser, 2003, Liu et al, 2003), process-oriented (Bastani, 2008), and behavioral (Heytmeyer, 2007). Each of these has its advantages and pitfalls.

A particular effort of linking together requirements engineering and modeling was taken by (Soares et al, 2008), applying a Model-Driven Requirements Engineering approach based on SysML Requirements and Use Case diagrams. The authors propose a process that includes classification for each atomic requirement after being written in natural language, and then using the SysML Requirements diagram for graphic representation of requirements and their relations.

Like UML, the SysML notation is comprised of multiple diagram types that provide mainly aspect decomposition. Hierarchical breakdown is supported for a subset of the diagram types. The multiple-view model of SysML makes it difficult to get a good grasp of the whole system, one of the major tenets of conceptual design.

Despite their inherent deficiencies, natural languages seem to be the only widely used and accepted means of requirement specifications. Requirement specification languages are a promising path to avoid ambiguity, but their disadvantages, including the time required to learn them, poor communication ability with non-technical stakeholders, and lack of widespread practice across domains, pose serious challenges to the user. The use of controlled languages, such as ACE - Attempto Controlled English0 (Kuhn, 2010) has not yet gained extensive spread either due to usability issues and low accessibility to non-experts.

Our approach differs significantly from the common practice in the following issues:

- We address the synthesis phase of requirements specification, providing a procedure for creating structured and unambiguous directives rather than analyzing existing, possibly confusing requirements.
- We use OPM as an underlying formalism, capitalizing on its advantage as a combined object-process method that addresses both structure and behavior in a single diagram and generates natural English sentences (Object-Process Language – OPL) on-the-fly in response to the user's graphic input.
- We build a procedure for constrained natural language representation (as opposed to techniques relying on more formal and less readable notation) on top of a strict logical backbone.

III. STRUCTURED REQUIREMENTS AUTHORING PRINCIPLES

Considering technical documents as products in their own right, the concept underlying the proposed methodology is applying the model-based systems engineering (MBSE) product development approach to technical documents. It relies on conceptual modeling, which precedes detailed document design, and extends it to developing and evaluating presentation alternatives and selecting the optimal one given its set of constraints .

The conceptual model of the document increases its reusability, providing the overall view of the document during its evolution. Not less importantly, a current conceptual model enables assessment of newly introduced or requested changes at the conceptual level, before engaging in expensive authoring activities. The cost of problems detected in text is prohibitively high, and it is worse thereafter. Such changes may cause expensive backtracking of relevant clauses throughout the text and further redesign of related paragraphs. The conceptual model is the ultimate instrument for detecting and solving problems starting at early stages of document development, when full text does not yet exist.

In order to provide means of communication amongst different, possibly non-technical stakeholders, the end result of the process has text as the main communication modality, side-by-side with its equivalent graphics-based model. The aims of our approach include (1) improvement of accessibility, as one can take the model that accompanies the document further to implementation, (2) increased inter- and intra-document consistency, as the modeling expresses the domain ontology, enhancing term definitions and creating links and hierarchies among domain concepts, and (3) provision of a tool for conformity and interoperation testing and evaluation of implementations.

To approach the authoring of technical documents in a systematic, structured way, we propose to treat the technical document as the end system to be delivered and the process of technical document authoring as a the function of the system that delivers the technical document. Just as the whole system is the outcome of the systems engineering process, the specification or standard is the outcome of technical document authoring process. Like any system, the technical document needs to be architected and designed prior to its production, i.e., its actual writing. Moreover, as a system, its complete lifecycle must be addressed.

Continuing the analogy between Systems Engineering and Technical Documents Authoring, a typical system undergoes during its life cycle phases of Specification, Design, Construction (in software: Implementation or Coding), Integration, Validation (in software: Testing and Debugging), Installation, Maintenance, and Retirement. These stages occur practically regardless of the system's lifecycle management methodology, possibly iterating and expanding several times, as in the spiral model. These stages are listed in the left hand column of Table 1.

Technical Document Authoring consists of the following stages: Document Definition, Document Design, Drafting, Checking, Maintenance, and Disposal. These are listed in the right hand column of Table 1 next to their counterparts in Systems Engineering, as explained next.

Viewing the technical document as a system in its own right, we obviously need to first state its 'raison d'etre', the main purpose of being. Specifically, we need to define for the document being authored its Scope, Goal, Objectives, Stakeholders, and Expected Benefit or Value for each stakeholder, possibly referencing to relevant prior material. Calling this stage Document Definition, we consider it to be the counterpart of Requirements Specification phase in a classical system lifecycle.

No.	System / Software Engineering Stages	Document Authoring Stages
1	Requirements Specification	Document Defining
2	Design	Document Designing
3	Construction (Implementation or Coding)	Document Drafting
4	Validation (Integration, Testing and Debugging)	Document Checking
5	Installation and Maintenance	Document Maintenance
6	Retirement	Document Disposal

Table 1: System Engineering and Requirements Authoring phases

Once Document Definition is done, the next process is Document Designing, which resembles the Design phase of a generic system. Commonly, this includes in-depth description of the main purpose of the document, its sub-processes, their results, and the attributes they require. This combines elements of Functional Decomposition, Abstraction, Basic Problems Definition and later stages of concept analysis.

Having the document designed, it is time for the Implementation phase, moving from blueprints to bricks and mortar, which, in our case, are words, sentences, and clauses. In the world of Technical Documents Authoring, we call this phase Drafting, and its outcome is the Technical Document Draft.

But have we got the Draft all right? Here is where Checking plays an important role by enabling us to check the outcomes of the Document Design phase with the outcomes of the Drafting phase.

Later, Maintenance of a Technical Document in use is akin to maintenance of a generic system until Disposal stage is reached. The Technical Document needs to be considered for updates stemming from such drivers as the ever changing stakeholder demands, new required capabilities, and new technologies. This requires constant change of the product (the text), and consequent change of the design, effectively yielding a development model that is equivalent to the Spiral Model in Systems Development Lifecycle.

IV. OBJECT-PROCESS METHODOLOGY (OPM) AND ITS USE FOR SUPPORTING MBRA

As argued, providing a general concept or guidelines is insufficient, as there are many formats and templates for documents. Instead, there is a need for some practical guidance.

We base our guiding method on Object-Process Methodology (OPM) 0(Dori, 2002), which offers a holistic approach, backed by a formal yet intuitive graphic and textual language, for model-based authoring of technical documents in general and requirements specifications in particular.

OPM is a framework for conceptual modeling of complex systems. It helps identifying essential objects and processes while minimizing ambiguity in functional and architectural requirements (Soderborg, 2003). This makes OPM suitable as a methodology and modeling language for the purpose of streamlining, formalizing, and explicating the specifications, and making them comprehensive, accessible, usable, and consistent both internally and externally.

The principles of OPM suit the need of gradual and consistent system presentation. An OPM model consists of a set of hierarchically organized Object-Process Diagrams (OPDs) that alleviate systems' complexity. Each OPD is obtained by in-zooming or unfolding of a thing (object or process) in its ancestor OPD. The top-level OPD, called the System Diagram (SD), contains the function of the system—its main process along with the enabling agents, instruments, inputs, and outcomes. In successively lower levels, SD is in-zoomed. In parallel, the corresponding objects are decomposed into their parts and specializations, and their features (attributes and operations) are gradually exposed.

Every diagram is limited to 20-25 things, catering to humans' cognitive limited capacity and following the principle of context maintaining or complexity management. For example, dealing with Space Exploring as a main process, manufacturing tolerances maintaining is not modeled. Similarly, welding cuts in the third stage of the fourth level engine of the emergency escape module are not modeled either. Instead, each process describes only things that directly affect it, while finer details are encapsulated in in-zoomed and unfolded diagrams.

One or more new things (objects and/or processes) can be specified within a thing in an OPD that was refined from a higher-level OPD. Copies of an existing thing can be placed in any diagram, where some or all the details, such as object states or links to other things, which are unimportant in the context of the diagram, can be hidden. It is sufficient for some detail to appear once in some OPD for it to be true for the system in general even though it is not shown in any other OPD.

OPM comprises entities and links. The three entity types are objects, processes (commonly referred to as "things"), and states. Objects are things that exist and can be stateful (i.e., have states). Processes transform objects: they generate and consume objects, or affect stateful objects by changing their states. Objects and processes are of equal importance, as they complement each other in the single-model specification of the system. Links, which are the OPM elements that connect entities, are of two types: structural and procedural. OPM objects relate statically to each other via structural relations, graphically expressed as structural links. The four fundamental structural relations are aggregation-participation, generalization-specialization, exhibition-characterization, and classification-instantiation. Objects can also be structurally related to each other by unidirectional or bidirectional tagged relations, similar to association links in UML class diagrams. Structural relations specify relations between any two objects. Due to the object-process symmetry, they can also specify relations between any two processes. Conversely, procedural links connect a process with an object or with an object's state to specify the dynamics of the system. Procedural links include (1) transforming links: effect link, consumption link, result link, and the input-output links pair, (2) enabling links: agent and instrument links, and (3) control links: event, condition, invocation, and time exception links.

Object-Process Diagrams (OPDs) are inherently accompanied with auto-generated OPL (Object-Process Language) text. OPL. According to 0(Dori, 2002)0, "… OPL is a dual-purpose language. First, it serves domain experts and system architects engaged in analyzing and designing a system, such as an electronic commerce system or a Web-based enterprise resource planning system. Second, it provides a firm basis for automatically generating the designed application".

V. GENERATING DOCUMENTS FROM OPM DIAGRAM

We have established a procedure for producing clear and less ambiguous documents by incorporation of OPM into the authoring process. This procedure was reviewed by learning lessons from authoring real documents.

Taking as an example the OPM-based documents authoring methodology described in the previous section, we will now demonstrate its principles with the aid of OPM modelling.

Below is the OPM model of the proposed OPM-based documents authoring methodology. It was created by examining the text in the above section and analyzing it by following OPM modelling conventions. For example, this forced turning all process names to gerund form and aligning the processes from top to bottom according to OPM timeline.

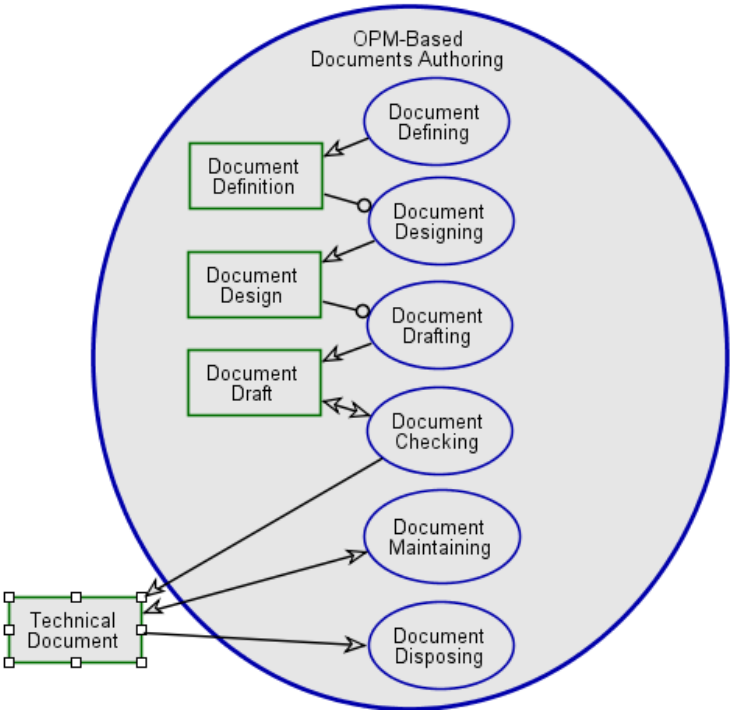

Figure 1: OPM-Based Documents Authoring

During the phase of modelling we have gained deeper understanding of the subject matter along with better distinction between central parts and irrelevant details that can be omitted or moved to subsequent diagrams. As a result, we have added more elaborate objects (like 'Document Design'), shifted objects like 'Scope' etc. to finer sub-levels, explicated the roles of the different items in the diagram and consolidated the overall hierarchy of the process and its attributes. Having this done, we can now automatically generate text that would describe the diagram in a coherent and structured way:

OPM-Based Documents Authoring process is comprised of **Document Defining**, **Document Designing**, **Document Drafting**, **Document Checking**, **Document Maintaining**, and **Document Disposing** processes.

Document Defining is the process of creating **Document Definition**, which is then used in **Document Designing** process for the sake of creating **Document Design**.

Document Drafting is the process of creating **Document Draft**, with the aid of **Document Design**.

Document Checking is the process of refining **Document Draft**, consequently creating the **Technical Document**.

Document Maintaining affects **Technical Document** along its lifetime, until it is discarded through **Document Disposing**.

VI. CONCLUSION

We have presented our research, addressing Requirement Documents as the scoped system under development, as part of a complete Model-Based Documents Authoring (MBDA) methodology. We intend to further develop the different MBDA aspects and employ it on several ISO standards. Hand-in-hand with developing the methodology, we build a graphical model-text editor to help the document authors—the system designers—to formally convey their thoughts and directives in a structured mode.

Applications of our approach can be diverse. These include linking Requirement Engineering and Project Management, possibly creating a connection between requirement models and a project plan. Another research and development avenue concerns knowledge management, capitalizing on the easier context-based accessibility of models compared with text-only files. This can be a new approach not only to requirement engineering and specification authoring, but also to technical documentation in general, relying on domain models and ontologies.

REFERENCES

Bastani, B., "Process-oriented abstraction of the complex evolvable systems: problem model construction", *ACM SIGSOFT Software Engineering Notes*, Volume 33 , Issue 3, May 2008.

Blekhman, A., Howes, D. B. and Dori, D., "Model-Based Verification and Validation of a Manufacturing and Control Standard", *2010 MSOM (Manufacturing and Service Operations Management Society) International Conference*, Haifa, Israel, June 27-29, 2010.

Dori, D., "Object-Process Methodology - A Holistic Systems Paradigm", Springer Verlag, Berlin, 2002.

Dori, D., Korda, N., Soffer, A. and Cohen, S., "SMART: System Model Acquisition from Requirements Text" *LNCS* 3080, pp. 179-194, 2004.

Dori, D., Martin, R., and Blekhman, A., "Model-Based Meta-Standardization: Modeling Enterprise Standards with OPM", *IEEE International Systems Conference*, San Diego, CA, USA, April 5-8, 2010.

Soares, M. and Vrancken, J., "Requirements Specification and Modeling through SysML", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics - SMC*, pp. 1735-1740, Montreal, Canada, 2007.

Grünbacher, P., Egyed, A., Medvidovic, N., "Reconciling Software Requirements and Architectures: The CBSP Approach", *5th IEEE International Symposium on Requirements Engineering*, Toronto, Canada, August 2001.

Heitmeyer, C. L.,"Formal Methods for Specifying, Validating, and Verifying Requirements", *Journal of Universal Computer Science*, vol. 13, no. 5, pp. 607-618, 2007.

IEEE: Software Engineering Committee of the IEEE Computer Society. IEEE Guide for Software Requirements Specifications (ANSI). IEEE Standard 830-1998.

Johnsson, C., "An introduction to IEC/ISO 62264", 2003. http://isotc.iso.org/livelink/livelink/fetch/2000/2489/Ittf_Home/MoU-MG/Moumg159.pdf , Accessed Nov. 12 2009.

Kasser, J. E.,"Object-Oriented Requirements Engineering and Management", *Systems Engineering Test and Evaluation (SETE) Conference*, Canberra, Australia, October 2003.

Kuhn, T.,"Controlled English for Knowledge Representation", *Doctoral thesis*, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, 2010.

Liu, Z., Jifeng, H., Li, X., and Chen, Y., "A Relational Model for Formal Object-Oriented Requirement Analysis in UML", UNU/IIST Report No. 287, October 2003.

Soderborg, N. R., Crawley, E., and Dori, D., "OPM-Based System Function and Architecture: Definitions and Operational Templates", *Communications of the ACM*, 46 (10), pp. 67-72, 2003.

BIOGRAPHY

Alex Blekhman (blekhman@tx.technion.ac.il) is a Ph. D. student at William Davidson Faculty of Industrial Engineering and Management at the Technion, Israel Institute of Technology, Haifa, Israel under the supervision of Prof. Dov Dori.

Prof. Dov Dori (dori@ie.technion.ac.il) is a Professor in the William Davidson Faculty of Industrial Engineering and Management at the Technion, Israel Institute of Technology, Haifa, Israel, and a research affiliate at MIT, Cambridge, MA. Dov Dori is a member of the IEEE and the IEEE Computer Society.