6TH ISRAELI INTERNATIONAL CONFERENCE ON SYSTEMS ENGINEERING
INCOSE IL 2011

Systems Engineering
Look Before You Leap

www.iltam.org/incose_il2011

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. This approach is guided by adopting general systems engineering principles to requirements authoring and can be extended and applied broadly for authoring various kinds of technical documents. Requirements specifications are first conceptually analyzed and designed on the basis of OPM model construction rules. Then, constrained natural text is derived and adjusted based on the model. The stages and the benefits of this approach are presented via an example of an Automated Teller Machine (ATM) system.

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. 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 of an ATM that illustrates the stages and the benefits of our method in a typical system definition scenario. 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.**שגיאה! מקור ההפניה לא נמצא** (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 system 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, although

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.

Despite their inherent deficiencies, natural languages seem to be the only widely used and accepted means of requirement specifications. Following.**שגיאה! מקור ההפניה לא נמצא**, one might consider the use of requirement specification languages 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. THE ENGINEERING OF REQUIREMENT AUTHORING

Ever emerging changes of operational environment, stakeholder needs, and technological environment require tedious writing and endless rewriting of the specifications; often resulting in cumbersome phrasing. Having technical documents, such as requirements documents and standards, prone to incompleteness and inconsistency because of their complexity is highly undesirable, because it can lead to lack of functionality, costly development cycle with much rework, and even potential hazard to users due to incomplete specifications.

Many organizations are engaged in producing requirement-like documents, but they stop there and do not design the system, let alone produce it. Rather, their deliverables are requirements specifications and other technical documents. The target systems or processes described in these specifications are not of direct interest to them. Among these organizations are defense institutions, standards organizations such as the International Organization for Standardization (ISO), main contractors and integrators of large projects.

To approach the authoring of requirement 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 requirement 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 Requirements Authoring, during its life cycle, a typical system undergoes phases of Requirements 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 (Estefan, 2008), possibly iterating several times, as in the spiral model. These stages are listed in the left hand column of Table 1.

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

Table 1: System / Software Engineering Stages vs. Requirements Authoring Stages

Requirements Authoring consists of the following stages: *Document Definition*, *Document Design*, *Drafting*, *Validation Maintenance*, and *Replacement*. 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 the main function of the system. 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.

Table 1: System Engineering and Requirements Authoring phases

Once *Document Definition* is done, the next process is *Document Design*, 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 is a crucial phase. Compared to Integrated, Customer

Driven Conceptual Design Method (Hari et al, 2002), it 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 Verification 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. 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. The top-level OPM diagram, 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.

Object-Process Diagrams (OPDs) are inherently accompanied with auto-generated OPL (Object-Process Language) text. OPL cannot be used, though, for specifications, as it is too mechanical and repetitive. According to 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".

Therefore, OPL was never meant to become a text to be read by a general audience. Based on these two goals, the OPL generation algorithm is based on DFS (Depth-First Search) 0 and Dori, 2001). Obviously, it is not suitable for human reading, although it may be good for code generation, for example (as all possible contexts of an item are brought together). OPL is too detailed, technical and is structured in a way which makes it difficult for humans to comprehend.

Given this, we define Tesperanto as a language alternative to OPL. Unlike the DFS specification of OPL, Tesperanto is defined and presented in a BFS (Breadth-First Search) style. An overall Tesperanto generation algorithm follows the next guidelines:

1. For each process:
    1.1. List the name and the definition of the process along with its outcomes.
    1.2. List the agents.
    1.3. List the instruments.
    1.4. List all other related items (conditions, etc.)

1.5. List the dynamic aspects (events, invocations etc.)

2. Align and rephrase the text according to a set of rules. For example, if a process has an instrument that is comprised of two object parts, add them parenthesized in the same line. If there is a list of, say, 5 or more instruments, consider splitting it into two groups with a sentence for each.

3. If there are more processes in the same level, repeat the above for each process. If not, in-zoom and repeat doing the above for the next level.

V. AN ATM EXAMPLE

We illustrate the principles of authoring documents in a structured way described above by the following example. Suppose that we are tasked with creating a specification for a new Automated Teller Machine (ATM). Its *Document Definition* is given below:

A consortium consisting of five banks operates an ATM system. Each bank holds many accounts, one for each customer. An account is accessible through a cash card. A customer can own one or two cash cards. The main process the ATM system is designed to carry out the execution of cash dispensing and depositing transactions. Each transaction refers to a specific account. Executing the transaction will result in one of the following two outcomes: it can yield a (successful) transaction or it can issue a denial notice.

Moving to the *Document Design* step, we need first to state the main function of the system, which fills its defined purpose. In this example it is Transaction Executing. Later on, we introduce the elements which activate or operate the system, along with the items required for the action. Then we note the environment elements with which the system interacts (accounts, cash, etc.). After some iteration with different stakeholders, the OPD in Figure 1 is agreed upon.

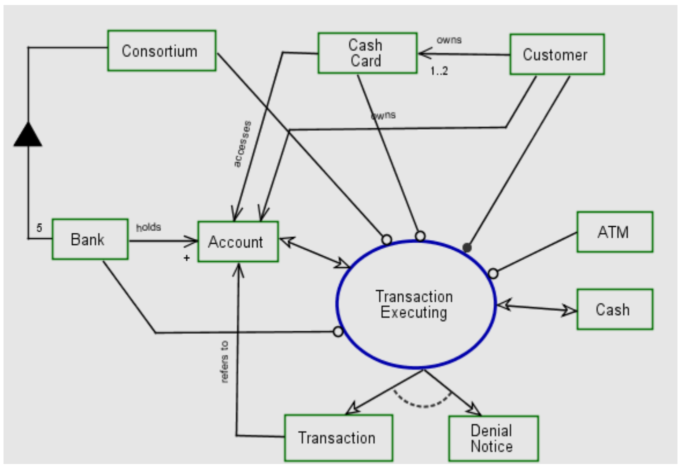

Figure 1: ATM Example OPM Model

Having the system depicted in a diagram enables us to critically review the model. While checking the system against the *Document Definition*, and possibly trying to express the model in text (i.e., making preliminary *Drafting* attempts), it appears that it is better to hide certain details and possibly move them to more detailed diagrams at lower-levels. The diagram in Figure 2 shows the system model after adjustment.

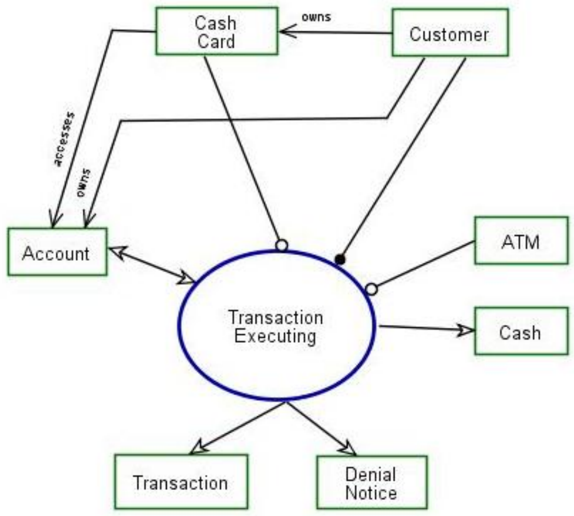

Figure 2: Adjusted ATM OPM Model

Continuing with the details, it is possible to zoom into the main process or unfold objects, which is not shown here. Instead, let us move along to the *Drafting* phase and see the text generated from the model. The color-coded generated text is shown below, where processes are in blue; objects are green, and remarks and explanations – bright grey):

Statics - Definition and things created:

Transaction Executing is a process that either performs a **Transaction** or issues a **Denial Notice**.

Agents and key instruments:

Transaction Executing is performed by a **Customer** with the aid of **ATM** and a **Cash Card**.

A **Customer** owns one or two **Credit Cards** and an **Account**, which can be accessed by each of the **Credit Cards**.

Dynamics:

Transaction Executing affects an **Account**, possibly drawing **Cash** from it.

We have not shown the further phases of *Validation* and *Maintenance* since these are straightforward, as soon as the principle of text coupling with the corresponding OPM model and mutual change is clear. The coupling is based on paragraphs consisting of simple, unambiguous sentences and their counterpart model fragments that convey exactly the same information. It inherently provides text-to-model consistency, such that making changes to the text triggers changes to the OPM model and vice versa, making the graphics and its associated text interchangeable and fully consistent at all times.

This procedure facilitates making technical documents humanly readable and writeable without compromising their formality, rigor, consistency and completeness following two major routes: backward and forward. In the backward direction, reviewing and improving existing specifications is performed, from their current text-based form to an OPM-based form that can be graphical and/or textual. In the forward direction, authoring new model-and-text-based specifications is performed according to the suggested scheme.

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". Berlin : Springer Verlag, 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", *2010 IEEE International Systems Conference*, San Diego, CA, USA, April 5-8, 2010.

Estefan, J. A., 'Survey of Model-Based Systems Engineering (MBSE) Methodologies', INCOSE-TD-2007-003-01, June 2008

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.

Hari, A., Herscovitz, J., Zonnenshain, A., Weiss, M. P., "Application of ICDM for the Conceptual Design of a New Product", *International Design Conference – Design 2002,* Dubrovnik, May 14 - 17, 2002.

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

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

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, 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.

Peleg, M., Dori, D., 'From Object-Process Diagrams to a Natural Object-Process Language, in *4th International Workshop on Next Generation Information Technologies and Systems*, Zikhron-Yaakov, Israel pp. 221—228, June 1999.

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 an associate 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.