This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Model-Based System Specification With Tesperanto: Readable Text From Formal Graphics

Alex Blekhman, Juan P. Wachs, and Dov Dori, *Senior Member, IEEE*

***Abstract*—Technical reports and papers may be represented by a fundamental model, which can take the form of a block diagram, a state-machine, a flow diagram, or alternatively some *ad hoc* chart. This basic scheme can convey better the true value of otherwise verbose and potentially encumbered narrative-based specifications. We present a model-based methodology for authoring technical documents. The underlying idea is to first formalize the system to be specified using a conceptual model, and then automatically generate from the tested and verified model a humanly-readable text in a subset of English we call Tesperanto. This technical documents' authoring methodology is carried out in an integrated bimodal text-graphics document authoring environment. The methodology was evaluated with the International Organization for Standardization standards and a medical robotics case study. The evaluation resulted in tangible improvements in the quality and consistency of international standards. Further, it can serve to document complex dynamics among agents, such as interaction between an operation room technician robot and the surgeon, suggesting that it could be applied to represent and bring value to other types of technical documents.**

***Index Terms*—Enterprise standards, medical treatment, modeling, object-process methodology (OPM), requirements, systems engineering (SE), technical documents.**

I. Introduction

Systems engineering body of knowledge defines systems engineering (SE) as "…an interdisciplinary approach and means to enable the realization of successful systems," and states that "…successful systems must satisfy the needs of their customers, users and other stakeholders" [1]. Accordingly, technical specifications, which constitute a central part of a systems engineer's work, must contain a clear, unambiguous, and complete representation of the required function, structure, and behavior of the system, product, or service in demand.

Manuscript received November 23, 2014; accepted January 11, 2015. This work was supported in part by the European Union's Seventh Framework Programme (FP7/2007–2013) under Grant 262044—VISIONAIR, and in part by the Gordon Center for Systems Engineering at the Technion. The work of J. P. Wachs was supported by the Qatar National Research Fund (a member of the Qatar Foundation) under NPRP Award NPRP6-449-2-181. This paper was recommended by Associate Editor M. P. Fanti. *(Corresponding author: Juan P. Wachs.)*

A. Blekhman is with the Technion—Israel Institute of Technology, Haifa 32000, Israel.

J. P. Wachs is with the School of Industrial Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: jpwachs@purdue.edu).

D. Dori is with the Technion—Israel Institute of Technology, Haifa 32000, Israel, and also with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2015.2406753

Growing system complexities have underscored the need for a natural and still formal approach of analyzing needs and articulating conceptual designs of novel systems, as well as stating technical specifications over a wide range, including systems' characterization, components, services, manufacturing and testing processes, operation guidelines, and technical manuals.

A figure of the level of complexity of common technical narratives is illustrated in Fig. 1. It presents the functional enterprise-control interface and related procedures, as a key component of the IEC 62264 Standard. Fig. 1 presents a large amount of components, displayed in a free and informal form, which makes nearly impossible maintaining the consistency. We conducted a detailed manual analysis which revealed numerous inconsistencies in the IEC 62264 current free-text standard specification. Some of these flaws were discussed in [2]. Also, early model-based systems specifying techniques were applied to indicate several cases of inconsistencies found in real-published standards (refer to [3]).

A. General Knowledge Representation Approaches

Knowledge representation and ontologies is a field of intensive research. Examples of these representations are medical dictionaries and taxonomies and software-related resources, (e.g., application-based domain modeling approach [4] and core components technical specification [5]). They specify high level rules and metamodels required to describe the fundamental structure and conceptual contents, processes, physical/logical data, and information exchange models.

In the field of system modeling techniques, we have examined UML [6], xUML [7], SysML [8], Modelica [9], object-process methodology (OPM) [10], and model-based dependability analysis [11].

UML and xUML were discarded as being software-oriented and not quite appropriate for modeling systems. In contrast, SysML was designed for modeling general systems [8]. The notation modifies a subset of UML diagrams and adds two new diagrams: 1) requirements diagram and 2) parametric diagram. Like UML, the SysML notation is comprised of multiple diagram types that provide mainly aspect decomposition [8]. 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, compromising one of the major tenets of conceptual design. Moreover, unlike xUML, the notation does not use formal activity specifications, and it allows free text in parametric diagrams for equation descriptions [7]. This informality makes it impossible to define SysML model-based execution without changing the notation significantly.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

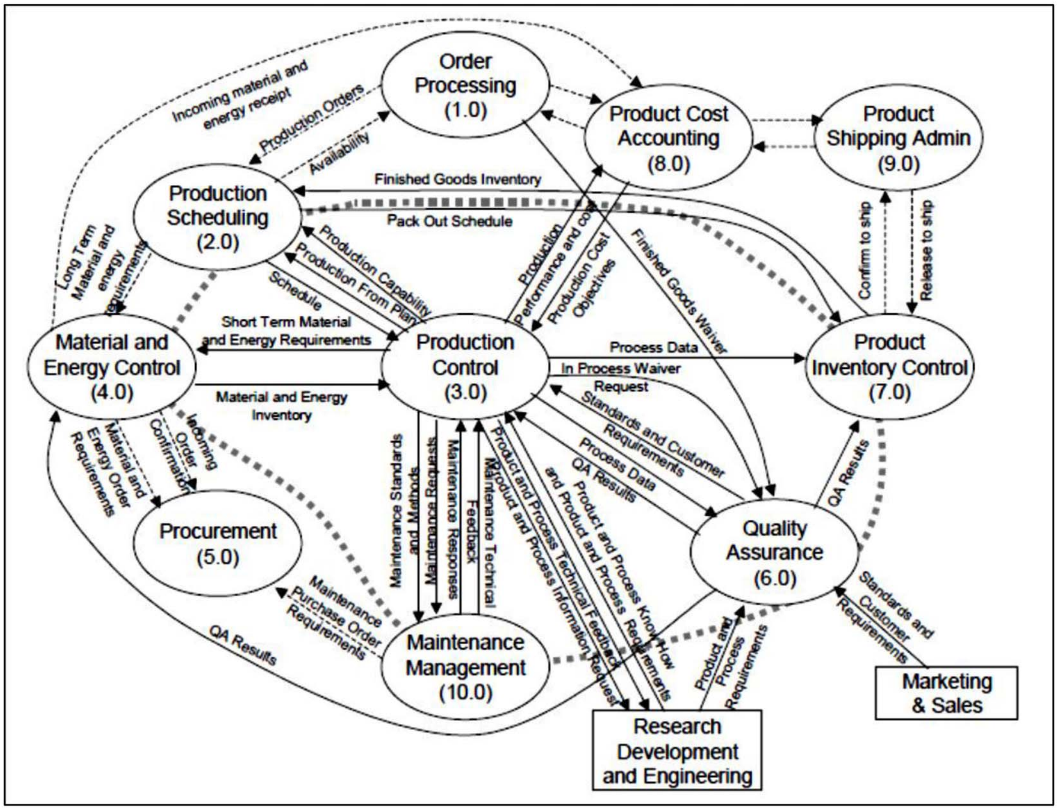


Fig. 1. Complex specifications in IEC 62264-1 functional enterprise-control model.

Modelica and MATLAB/Simulink are most fit for the detailed design phase [9]. Indeed, Simulink has been combined with OPM to perform computations within an OPM model [12], but these languages are too technical for the conceptual modeling phase and for use by a wide user community.

B. Current Specification Modeling and Requirement Engineering Techniques

Specifications and requirement statements are 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. For example, IEEE Standard 830-1998, 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” [13, Sec. 4.3, p. 4]. While these principles appear to be intuitive and concise, summarizing years of common experience, it is often difficult to translate them into practical implementation, as the requirements engineer is given very little, if any, concrete guidance, tool, or procedure.

Representation tools, requirement methods, and languages can be categorized into object-oriented [14], [15], process-oriented [16], and behavioral [17]. Each of these has its advantages and pitfalls. Yet, current representation schemes of technical requirements are not mature enough to cope with problems of consistency and other deficiencies described above. Suffering from ambiguity and lacking a clear and direct translation of customer needs/wants into precise specifications, the early lifecycle stages of most project-product systems are by and large incoherent and unstructured.

C. Existing Computer-Aided Tools

Several tools and environments for visual modeling or programming exist. Graphic editors, such as Microsoft Visual Studio (http://msdn.microsoft.com/vstudio/) and web design tools, such as Adobe Dreamweaver (http://www.adobe.com/products/dreamweaver/) allow simultaneous text and graphics editing. Metafor [18] and other text-to-code environments provide functionality for dual text and code viewing, scripting, and editing.

Visual paradigm for UML (http://www.visual-paradigm.com/product/vpuml/) provides means for manual editing of use cases with a limited text analysis utility that allows manual markup and extraction of keywords from text.

Numerous works (see [19], [20]) discuss transforming natural language into code. Natural language processing approaches and tools are often used as aids in these tasks.

Other works have previously addressed text generation from relational databases and mathematical documents.

Fakas [21] concerns with the generation of a complete and consistent set of information from data that contains cross-links and relations. The authors use label lists in order to deal with the intersections and introduce schema compliant mechanisms with certain regulations and formats.

Barzilay and Lapata [22] and Barzilay and Lee [23] introduced various methods for text-to-text generation using automatically-induced models of text structure and other constructs. While their methods constitute state-of-the-art approaches, she rightfully states that "... most concept-to-text generation systems operate in limited domains, using a set of deterministic rules is a feasible way to capture patterns in content organization." This is also the general path of our research.

O'Donnell and Knott [24] stated that "... A database by itself does not provide enough information to produce text. Domain semantics is required." This gap is addressed in our case with Tesperanto construction rules.

The development of technical documents often involves adding pictorial and diagrammatic references to the text's narrative. Yet, the main modality of expression is the written word, and we firmly believe will remain. With this in mind, this paper was set to enhance the quality, coherence, and completeness of technical reports and documents by modeling the implicit scheme supporting the technical document narrative. Instead of interpreting the free textual content with its respective figures and informal charts, the text will be created from the recovered model. The integration of an core model and text associated with that model and reflects it textually allows a clear presentation of the document's content, offering an analytical demonstration of the author's intention. Our claims is that building on formal model as a fundamental step can enable the rendering, authoring, and analysis of documents in a much simpler and unencumbered way. Such a model can act as a consistent, solid, and unambiguous basis for analyzing the system, comparing it to models of other systems, and even generate code for the relevant software-intensive portions of the system.

In the same vein of the model-based SE (MBSE) approach and the ideas previously discussed, the first version of Technical Esperanto (Tesperanto) methodology and language was created.

The rest of this paper is organized as follows. Section II introduces the principle involved in model-based system specification (MBSS) and Tesperanto, along with an application to requirements specifications. Section III discusses the validation of the aforementioned paradigm through industrial, medical, and the International Organization for Standardization (ISO)-related case studies. Section IV discusses the conclusion.

II. Principles of MBSS

A. *Specifying Goals of the Model-Based Systems and the Fundamental Formalism*

The model-based systems specifying methodology (and subsequently Tesperanto) was originated by ISO. ISO standards, as well as those of other standardization 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 [25].

We claim, that the major root cause for this state of affairs is that technical documents in general, and standards in particular, are built with no comprehensive base reference with which to measure the coherence of the technical content they convey. Technical documents are expressed as clauses or statements in free text, possibly accompanied by figures that sometimes do not match the text or conflict with other figures. The lack of an underlying analytical process that would accompany the creation and maintenance of technical documents, such as requirement specifications, directives, and standards, and ensure their quality, is a serious limiting factor. This process deficiency also limits the length and complexity of a comprehensible document.

Therefore, through the development of Tesporanto, we try to expose and work with the actual scheme underlying the technical document text, rather than interpret clauses or statements in free text, possibly accompanied by disparate figures and mostly informal diagrams that sometimes do not match the text or conflict with other figures. We therefore suggest, that converting textual information to a model-based structured form helps pinpoint missing elements and resolve ambiguities. The combination of model and text should allow for the document content to be presented in an unequivocal manner, providing an analytical representation of its author's intent.

The Tesperanto ontology represents a mapping from text to concrete concepts and relations. It provides practical guidance for the following OPM principles that ensure consistent system presentation and gradual exposure of details.

The modeling framework—both the methodology and the language—underlying the Tesperanto approach must be general enough to cater to a wide variety of domains and simple and intuitive enough to be used by experts from various fields. Further requirements include expressiveness of the notation for defining common conceptual models, a high level of formalism, and clear, unambiguous, and nonredundant semantics. The concept of the modeling framework we built follows general SE approaches that emphasize the dynamic aspects of systems, such as soft systems methodology [26] and structural modeling technique [27].

Following the survey of knowledge representation approaches in [3], which included concept maps, semantic networks, conceptual graphs, knowledge interchange format, the common logic standard initiative, and OPM, we stick to the observations made there in favor of OPM and it was selected as the underlying conceptual modeling language for Tesperanto. The main features of OPM that support the selection are as follows.

1) OPM is a visual methodology that incorporates the static-structural and dynamic-procedural aspects of a system into a unifying model, which is presented in its entirety using a single diagram type. This is achieved by treating both objects and processes as equally important things (entities). By using a single model at varying levels of detail, clutter, and incompatibilities are likely to be avoided even in highly complex systems.
2) OPM is designed to express triggering events, guarding conditions, timing constraints, timing exceptions, and flow-of-control constructs. These features are the basic elements required for exceptional behavior design.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
SYSTEM ENGINEERING AND REQUIREMENTS AUTHORING PHASES

No.	System / Software Engineering Stages	Document Authoring Stages
1	Conceiving & 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 Maintaining
6	Retirement	Document Disposing

3) OPM has proven to be an efficient methodology for modeling complex dynamic behavior patterns in general and temporal exceptions in particular.
4) OPM was shown to be significantly better in specification quality, compared with object modeling technique and UMLs main predecessor.

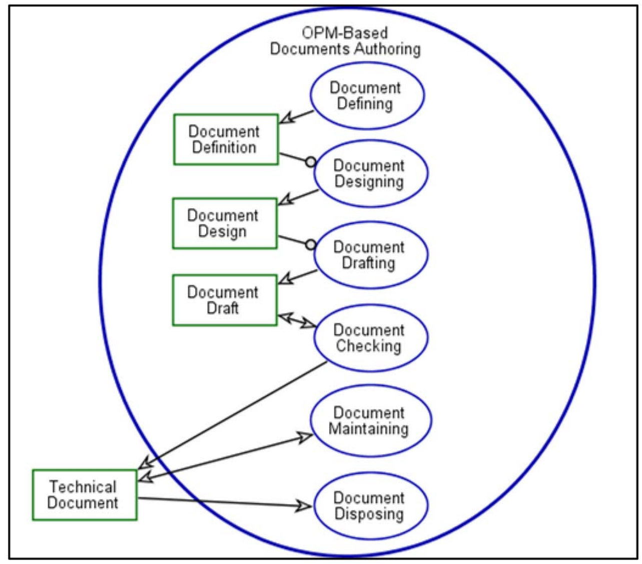


Fig. 2. OPM-based documents authoring model.

B. Document-as-Product Principle

Following [3], we propose to treat the technical document as the end system to be delivered and the process of technical document authoring as the function of the system that delivers the technical document. This approach renders authoring of technical documents systematic and structured. Treating the document as a system, we apply known SE techniques (as brought, see [28]) to the domain of technical writing [3]. Just as the whole system is the outcome of the SE process, the specification or standard is the outcome of the 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 SE and technical documents authoring, during its lifecycle, a typical system undergoes phases of conceiving and requirements specification, design, construction, (implementation or coding in software systems), validation (integration, testing, and debugging in software systems), installation and maintenance, and retirement. These stages, listed in the left column of Table I, occur regardless of the system's lifecycle management methodology, possibly iterating and expanding several times, as in the spiral model. Analogously, technical document authoring consists of document definition, document design, document drafting, document checking, document maintenance, and document disposal. These are listed in the right column of Table I, next to their SE counterparts, and modeled in Fig. 2.

Viewing the technical document as a system in its own right, we obviously need to first state its "raison d'être," the main purpose for its existence—the value it delivers via its function. 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 defining, we consider it to be the counterpart of requirements specification phase in a classical system lifecycle.

Document designing, which follows document defining, resembles the design phase of a generic system. Document designing combines elements of functional decomposition, abstraction, basic problem definition, and later stages of concept analysis.

Having designed the document, it is time to move from blueprints to bricks and mortar in the implementation phase. In our case, the bricks and mortar are words, sentences, and paragraphs or clauses arranged in a hierarchy. In the domain of technical documents authoring, we call this phase the document drafting process, and its outcome is the object technical document draft.

But have we gotten this technical document draft all right? Here is where document checking plays an important role: it enables us to check the outcomes of the document drafting phase and compare them to the outcomes of the previous document designing phase, yielding the technical document as the product to be delivered.

Once delivered and used, document maintaining of our technical document that is now in use is akin to maintenance of a generic system, until the document disposing 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. These require constant change of the document design and consequent change of the product—the document itself, effectively yielding a development model that is equivalent to the spiral model in a system development lifecycle.

Based on this "document-as-a-product" idea, we have produced an OPM model of a document authoring system whose function is the process of producing technical documents. This model was put to test iteratively reviewed and improved by lessons learned from authoring real-life sizeable technical documents.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

The **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 the **Document Designing** process for the creation of **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**

Fig. 3. Tesperanto paragraph for OPM-based documents authoring process.

Consider the following "meta-example" expressed in the OPM model of the proposed OPM-based documents authoring methodology. This model was created by examining the text in the above section and analyzing it by following OPM modeling conventions. Among other outcomes, the OPM modeling process has encouraged turning all the process names to their gerund form and aligning the processes from top to bottom according to their timeline within the context of the in-zoomed OPM process called OPM-based document authoring.

During the modeling phase, 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 lower-level diagrams. As a result, we have added more elaborate objects (like document designing), we moved objects, such as scope, to lower-level diagrams, made the roles of the different items in the diagram more explicit, and consolidated the overall hierarchy of the process and its attributes. Using this improved model, we have automatically generated the Tesperanto text in Fig. 3.

C. Model-Based Systems Specifying Process

The model-based systems specification (MBSS) methodology and tool set consist of the Tesperanto ontology and language, MBSS process guidelines and document templates, and the model-based authoring of specifications environment (MBASE) software package.

The MBSS development process is modeled in Fig. 4. It is corresponding Tesperanto paragraph is brought in Fig. 5. This paragraph was automatically generated from the model in Fig. 4 by the MBSS methodology itself and the tools it describes.

D. Model-Based Systems Specifying Principles

The objective of the MBSS methodology is creating consistent and well-structured specifications either from scratch or by analyzing existing material. A typical MBSS-based workflow includes the following steps.

1) Creating an OPM model of the technical content to be documented, either as new material or based on some existing textual specification, possibly with accompanying graphics.
2) Automatically generating a Tesperanto text skeleton from the OPM model using MBASE.

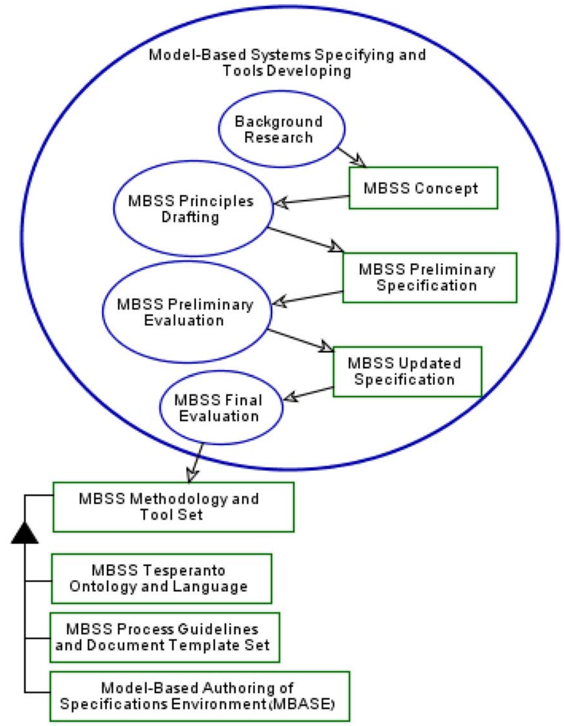


Fig. 4. MBSS and tools developing.

The **Model-Based Systems Specifying & Tools Developing** process is comprised of the following four subprocesses: (1) **Background Research**, (2) **MBSS Principles Drafting**, (3) **MBSS Preliminary Evaluation**, and (4) **MBSS Final Evaluation**.
Background Research is the process of creating a **MBSS Concept**.
MBSS Principles Drafting is the process of creating a **MBSS Preliminary Specification**, based on **MBSS Concept**.
MBSS Preliminary Evaluation is the process of creating a **MBSS Updated Specification**, based on **MBSS Preliminary Specification**.
MBSS Final Evaluation uses **MBSS Updated Specification** in the process of creating a **MBSS Methodology & Tool Set**, which consists of a **MBSS Process Guidelines & Document Template Set**, a **Model-Based Authoring of Specifications Environment (MBASE)**, and a **MBSS Tesperanto Ontology & Language**.

Fig. 5. Tesperanto paragraph for model-based systems specifying and tools developing process.

Welding two steel parts together creates the combined part.

Fig. 6. Basic welding process specification.

3) For an existing document, replacing its free text specification with corresponding newly created Tesperanto specification and its corresponding OPM model.
4) If and where required, manually adjusting the Tesperanto text skeleton to form a structured specification. MBASE tracks changes and assures that the edited text remains consistent with the model at all times.

To illustrate the application of MBSS, consider the following simple process specification in Fig. 6. This sentence is presented as the object-process diagram in Fig. 7. Using MBASE to convert the model in Fig. 7 back to text, we get the Tesperanto paragraph in Fig. 8. If, for example, it is important to specify the fact that a gas metal arc is needed for this process, the model will change accordingly, as shown in Fig. 9.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

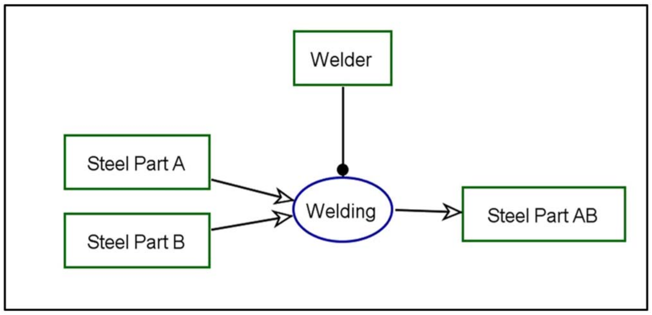


Fig. 7. OPD for a basic welding process.

> **Welding** is the process of creating a **Steel Part AB** by a **Welder**. This process consumes a **Steel Part A** and a **Steel Part B**.

Fig. 8. Tesperanto paragraph for a basic welding process.

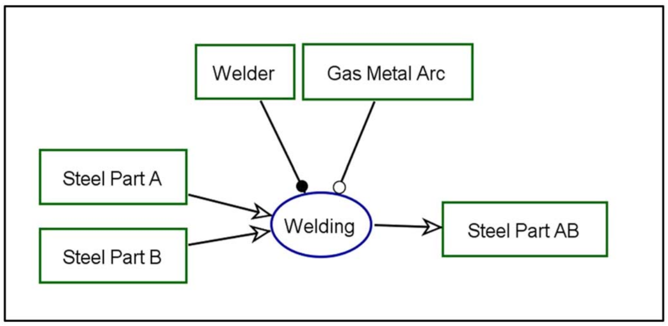


Fig. 9. OPM diagram for welding with a gas metal arc process.

> **Welding** is the process of creating a **Steel Part AB**, with the aid of a **Gas Metal Arc**. This process is performed by a **Welder**, consuming a **Steel Part A** and a **Steel Part B**.

Fig. 10. Tesperanto paragraph for welding with a gas metal arc process.

The Tesperanto text corresponding to Fig. 9 will automatically reflect the introduced change, as expressed in the text in Fig. 10.

A more realistic example of the application of the stated above principles was brought in [3], where a part of IEC 62264 Standard was modeled. Consider the source specification in Table II, which illustrates some typical problems of combining free text with graphic specifications.

1) *Inconsistency Between Figure Notation and Notation in Text:* For example, specific personnel (text of this paragraph) versus person (in the model and later in the text) or qualifications of personnel (in the text of this paragraph) versus person property (in the model and later in the text).
2) *Incomplete Text (Information in the Model is Not Present in the Text):* For example, the relation "records the execution of" between qualification test result and person property.
3) *Incomplete Figure (Information in the Text is Not Present in the Figure):* For example, correspondence to ISO 15704 and ISO 15531-1.

Only a few of these issues are resolved later in the standard's text, while the majority must be inferred from context.

TABLE II
CORRESPONDING TEXT AND FIGURE FRAGMENT OF PARAGRAPH 7.5.1.1—PERSONNEL MODEL OF ISO/IEC 62264

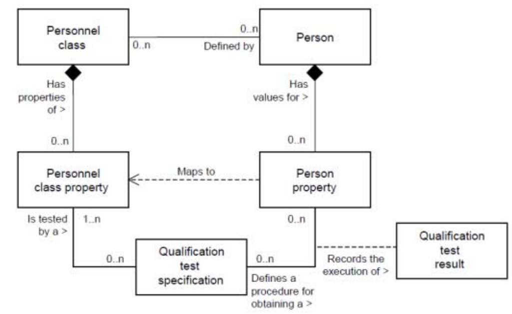
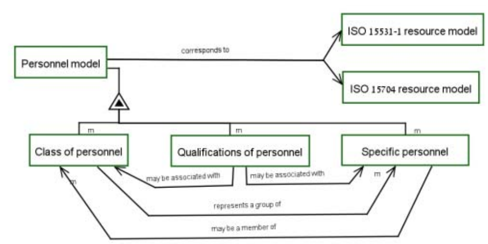
Text	The personnel model contains the information about specific personnel, classes of personnel, and qualifications of personnel. Figure 14 illustrates the personnel model. This corresponds to a resource model for personnel, as given in ISO 15704 and ISO 15531-1
Figure	 Figure 14 – Personnel model

TABLE III
MODEL-BASED VERSION OF IEC 62264 EXCERPT IN TABLE II

Text	**Personnel model** corresponds to **ISO 15704** and **ISO 15531-1 resource models** and contains the information about **Specific personnel**, **Qualifications of personnel** and **Classes of personnel**: **Class of personnel** represents a group of many **Specific personnel**. **Specific personnel** may be a member of many **Classes of personnel**. **Qualifications of personnel** may be associated with **Class of personnel** and/or with **Specific personnel**.
Figure	

We converted the text and figure for IEC 62264 paragraph 7.5.1.1—personnel model (see Table II) to OPM model and Tesperanto text. The resulting model and paragraph are shown in Table III. Now, the text conveys complete information in a consistent form and is fully aligned with the model. The text is composed of simple, light, and unambiguous sentences that, in addition to its simplicity and explicit nature, are also likely to significantly facilitate automated, yet reliable, translations to natural languages other than English. Further detailed information, which causes confusion in the original specification, is hidden in in-zoomed paragraphs for specific personnel, qualifications of personnel and classes of personnel.

E. Tesperanto Overview

The MBSS methodology, and Tesperanto as part of it, follows MBSE principles and is based on corresponding and simultaneous bimodal representation of the OPM model with text and graphics [10]. A cornerstone of the MBSS methodology is the Tesperanto language.

Tesperanto is designed as a language alternative to, and on top of OPL. OPL is too detailed, technical, mechanical, and repetitive. It is structured in a way that makes long text difficult for humans to follow due to repetition of basic constructs. According to [10], "…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."

Indeed, OPL was not explicitly meant to become a text to be read by a general audience. The OPL generation algorithm is based on depth-first search (DFS). It is not natural for human reading, although it may be good for lower-level machine-oriented tasks, such as code generation.

Unlike DFS specification of OPL, Tesperanto is defined and presented in a breadth-first search (BFS) style. Tesperanto absorbs OPMs gradual presentation principles catering to humans' cognitive limited capacity and following the guidelines of context maintaining or complexity management.

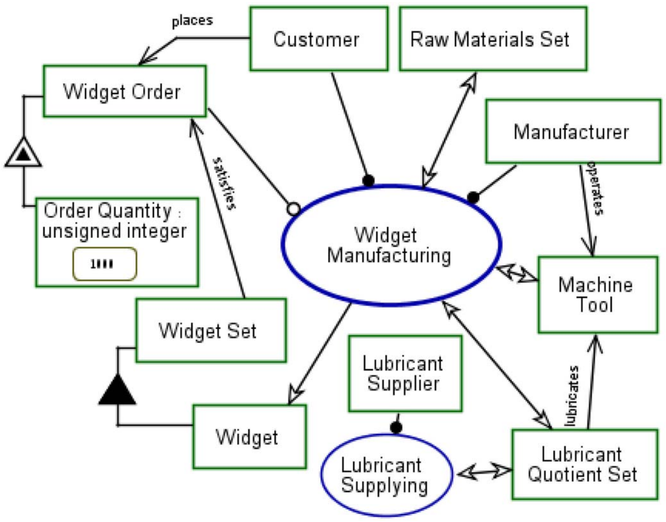


Fig. 11. OPM diagram for widget manufacturing process.

Widget Manufacturing is a process of manufacturing **Widgets** according to **Widget Order** that is placed by a **Customer**. **It** is performed using **Raw Materials Set** by a **Manufacturer** that operates **Machine Tool**. […]

Fig. 12. Tesperanto paragraph for widget manufacturing process.

Customer places **Widget Order.**
Customer handles **Widget Manufacturing.**
Manufacturer operates **Machine Tool.**
Manufacturer handles **Widget Manufacturing.**
Lubricant Quotient Set lubricates **Machine Tool.**
Lubricant Supplier handles **Lubricant Supplying.**
[…]

Fig. 13. OPL paragraph for widget manufacturing process.

Fig. 14. Gestonurse delivers a surgical instrument to the main surgeon.

F. Tesperanto Generation Algorithm

The basic principle of Tesperanto language is that it is model-based. That is, each phrase has its graphic counterpart as a model fact. Therefore, each sentence, phrase, or token can be traced down to its representation in the model.

Another fundamental property of Tesperanto is its graphical model-text consistency. A change of some Tesperanto text is immediately followed by an appropriate corresponding change of the OPM graphical model. Likewise, a model change is immediately reflected in the Tesperanto text.

The Tesperanto generation algorithm is based on a set of ordinal and layout rules. Ordinal rules prescribe the order of appearance for model elements in text. Following OPM functional decomposition practice [28], the focus is on the central process and its outcomes, e.g., the first sentence in Fig. 10 ("welding is the process of creating a steel part AB"). Next, the text regarding agents, instruments, and all other model facts is generated according to subsequent ordinal rules.

Tesperanto layout rules control overall paragraph structure. These rules put both minimum and maximum constraints on sentence length and the number of sentences in paragraph. Typically, a sentence is limited to 3–5 model facts, and up to two levels of dependence. For example, a sentence like "*X* is a process of creating *Y*, which includes *Z*, which is an instance of *W*" is illegal, as it has three levels of dependence, since *W* relates to *Z* which relates to *Y*, which in turn relates to *X*. On the other hand, in case of a short sentence that contains only one model fact, it may be united with the next phrase.

By default, Tesperanto follows a BFS presentation style. Informally, this results in text that relates first to the close links of a thing (object or process), and only then moving to farther things, as illustrated in Fig. 11 and the Tesperanto text following it.

The Tesperanto paragraph excerpt in Fig. 12 corresponds to Fig. 11, showing the gradual, breadth-first style of presentation.

For the sake of comparison, an OPL paragraph excerpt in Fig. 13 also corresponds to Fig. 11. The DFS definition of OPL and one-model-fact-at-a-time short, repetitive, and mechanical sentence style becomes apparent especially when compared to the Tesperanto text above.

Customization is another principle of Tesperanto, as it would be difficult to precisely predict the intent of the modeler at all times. The compact set of OPM elements (entities and links) does not allow for an exact translation algorithm that will produce the most appropriate text for systems in all the conceivable domains.

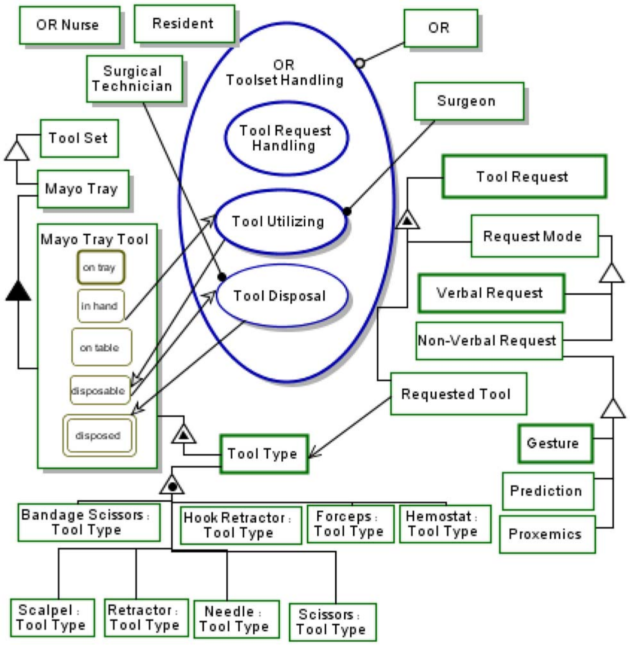


Fig. 15. OR toolset handling—source diagram used as a starting point.

The solution for this issue is the introduction of domain-specific templates that grasp common and recurrent phrases, idioms, etc. Hand-in-hand with using these domain-specific templates, the user has the ability to change the suggested wording, switch word order, etc., allowing multiple wordings and synonyms for the same model fact, as long as the link to the model fact remains. This text manipulation option allows the user to pick up the wording that best captures the intent while still being compatible with all the model facts expressed in the diagram. For example, in Tesperanto excerpt for Fig. 11, the default phrase "is a process of creating widgets" was manually adjusted to "is a process of manufacturing widgets." Alternatively, while trying to edit the text, the user might conclude that the model, rather than the text, needs to be edited, and MBASE supports this graphics-to-text direction as well.

III. MBSS and Tesperanto Assessment and Evaluation

A. Gestonurse Case Study

Operation room (OR) toolset handling is a process that is currently performed by a human OR technician. This process involves a number of activities related to using, delivering, and retrieval of surgical instruments during a surgical procedure. The most common of these activities is the delivery of the instruments. Understaffing, miscommunication, and lack of experience is often the root of mistakes occurring during this repetitive, yet necessary, task [29]. These mistakes can have negative outcomes, such as unnecessary delays, injuries due to improper delivery of sharps, and retained instruments [30].

Gestonurse is the first attempt to support the surgical team with a robotic scrub nurse (see Fig. 14) through the delivery of surgical instruments in a timely, accurate, and safe fashion [31]. In order to resemble human-only teamwork, both verbal and nonverbal human-robot communication forms must

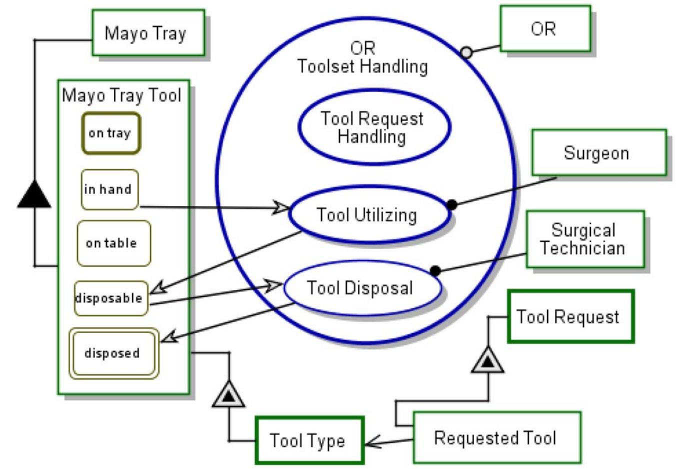


Fig. 16. OR toolset handling—reviewed diagram.

OR Toolset Handling is comprised of the following three subprocesses: (1) **Tool Request Handling**, (2) **Tool Utilizing**, and (3) **Tool Disposal**.
OR Toolset Handling is performed with the aid of **OR**.
Tool Utilizing is a process that results in turning in hand **Mayo Tray Tool** to disposable **Mayo Tray Tool**. This process is performed by a **Surgeon**.
Tool Disposal is a process that results in turning disposable **Mayo Tray Tool** to disposed **Mayo Tray Tool**. This process is performed by a **Surgical Technician**.
Mayo Tray Tool is part of **Mayo Tra**y and is characterized by **Tool Type**.
Tool Request is characterized by **Requested Tool** that is related to **Tool Type**.

Fig. 17. OR toolset handling—Tesperanto paragraph.

be employed for operating the robot and working with it as a team. While taxonomies have been suggested for verbal communication in the OR [32], there is a gap in the understanding of nonverbal forms of communication [33] and how they shape the instrument handling task in the OR. For example, it is not known how frequently surgeons request instruments using hand gestures, and how important is their body language, e.g., bending toward the patient versus standing straight, what role does the surgeon's gaze play in the type of instrument requested, and how often does the surgical technician predict the next instrument, before the surgeon actually requests it.

To answer these questions, a conceptual model of the communications in the OR regarding instrument handling was developed. The goal of this OPM-based conceptual model is to determine the necessary forms of communications required to interface with the robotic scrub nurse. A system of this nature may result in fewer mistakes, reduced miscommunications, and shorter procedure times [31]. The source object-process diagram of this process is shown in Fig. 15.

Following MBSS principles of analysis and modeling, and Tesperanto text generation, the diagram in Fig. 16 was eventually created. The main changes with respect to the original model were the elimination of the "weakly tied" objects tool set and Mayo tray (not used elsewhere), request mode and subsequent objects (elaborated in dedicated views—views 2 and 3), and tool type instances (elaborated in view 5). The corresponding Tesperanto paragraph appears in Fig. 17.

As reported by the group of colleagues working on the project, the use of OPM modeling and MBSS through

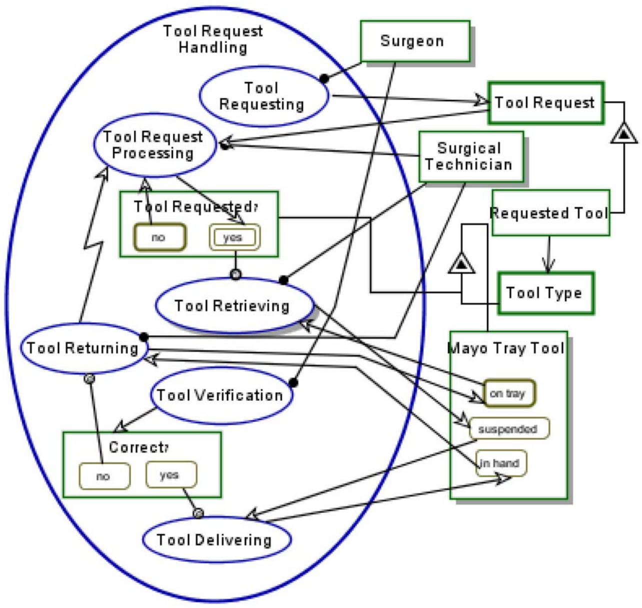


Fig. 18. Tool request handling—before applying MBSS.

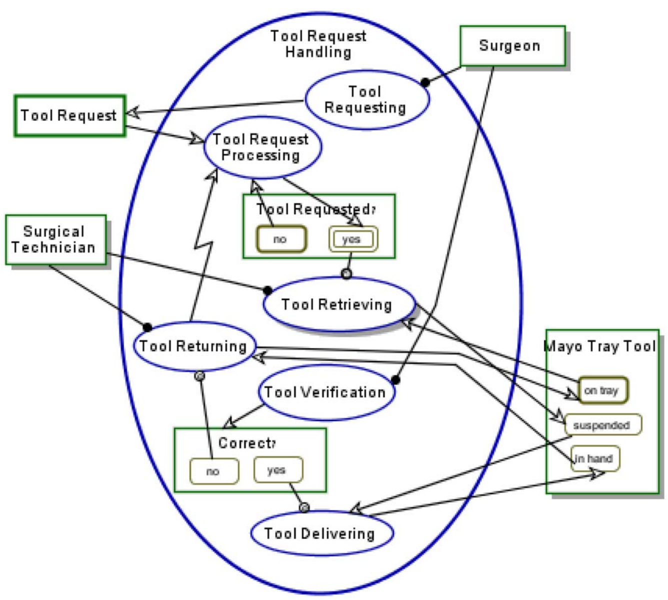


Fig. 19. Tool request handling—after one iteration of applying MBSS.

Tesperanto in the Gestonurse case study dramatically increased the quality of the specification, eliminating unnecessary details and focusing on a single, consistent, and unifying frame of reference for the entire system. These benefits are illustrated by examining the models of the tool request handling process before the change (Fig. 18), over one iteration of applying MBSS and Tesperanto (Fig. 19) and the final result (Fig. 20). As the project team noted, iteratively applying MBSS in this case has increased the consistency, clarity, accuracy, and eventually the correctness of the specification along with elaboration of the subject matter with domain experts, the model eventually converged to its (currently) final version, with which all stakeholders are satisfied with regard to scope, content, and expressiveness. As an illustration, the resulting model of the tool request handling process is shown in Fig. 19.

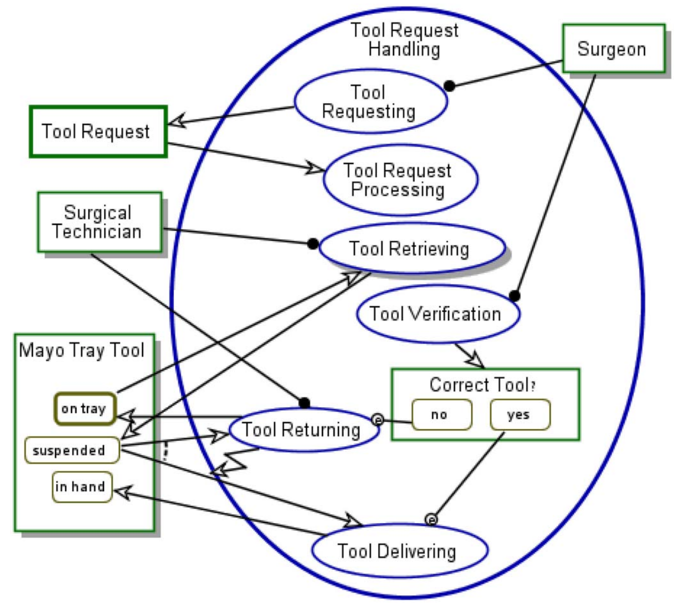


Fig. 20. Tool request handling—final model.

Data acquisition: Data acquisition modules collect and export production data and laboratory data, which form unified process data documents supporting soft sensor and craft calculation modeling module, and providing data for data-driven model and mechanism model.

Soft sensor modeling: Soft sensor modeling modules analyze data provided by data acquisition modules under off-line circumstance. Selecting soft sensor model combining the process requirements, calculating model parameters by parameter optimization, comparing the simulation model, the soft sensor modules provide the model for on-line management module as the basis of real-time calculation.

Soft sensor design: Soft sensor design modules are tools based on the process requirements, which can configure soft sensor model or craft calculation model.

On-line management: On-line management modules are universal management platform and management software for on-line engineering and backstage control service.

Laboratory correction: Laboratory correction is a special soft sensor module, which belongs to on-line module. The modules on-line revise soft sensor calculation outputs or models based on laboratory data.

Web server and network system: Web server and network system provides network monitoring, operational data service, and support on-line management, Web client communication and client rights management.

Web client: Web client provides access and operation of advanced process control system for remote network client, with the same function as on-line management.

Fig. 21. ISO 15746-1 pp. 19–20: 8.1 soft sensor—source specification (text).

B. ISO Expert Evaluation

The work on the MBSS methodology started following a presentation of model-based standard authoring with OPM at the ISO TC184/SC5 2009 annual meeting in Paris [12]. In Resolution 611 (Paris 21), ISO TC184/SC5 established the OPM Study Group in order to explore the usefulness of OPM for "…creating, designing, analyzing, and simulating models of its standards to improve the development, communication, and understanding of these standards."

Analyzing existing standards with this MBSS methodology has exposed inconsistencies, mismatches, missing, and

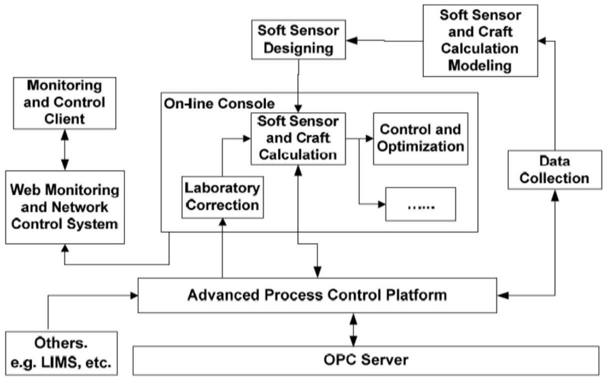


Fig. 22. ISO 15746-1 pp. 19–20: 8.1 soft sensor—source specification (diagram).

The Soft Sensing and Craft Calculating process is comprised of the following two subprocesses: (1) Laboratory Analyzing and (2) Laboratory Correcting. Furthermore, it operates in coordination with external Soft Sensor Designing, and Monitoring and Controlling processes.
Soft Sensing and Craft Calculating is performed according to an Advanced Process Control Platform Software System, which interacts with an OPC Server and may be instantiated by a LIMS. This process is performed according to Soft Sensor & Craft Calculation Model Design and consumes Process Simulation Data.
Laboratory Analyzing is the process of creating an Analyzed Data, consuming Measured Variables (Production Data).
Laboratory Correcting is the process of creating Analyzed Variables, with the aid of external Laboratory Data. This process consumes Analyzed Data.

Fig. 23. Soft sensing and craft calculating specification text (based on ISO 15746-1).

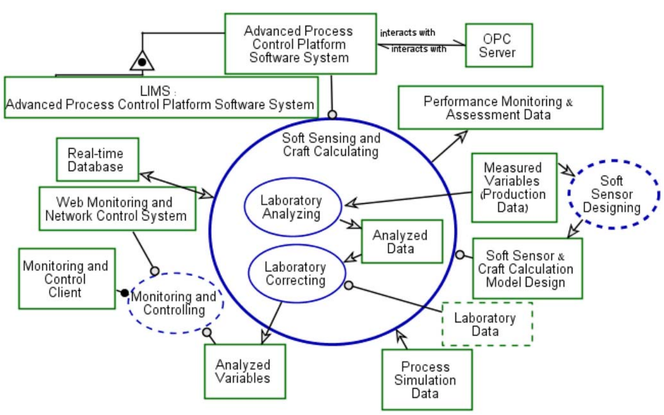


Fig. 24. Soft sensing and craft calculating specification model (based on ISO 15746-1).

confusing definitions between figures and text. Some of the documents examined were approved technical standards that are in current use, emphasizing the importance of streamlining technical specifications in a consistent, standard manner.

As an example of employing MBSS for specifying ISO standards, Figs. 20 and 21, respectively, present the text and the accompanying diagram for a soft sensor specification in ISO 15746-1.

By applying MBSS and Tesperanto on the specification in Figs. 21 and 22, numerous internal and external inconsistencies were noted. The resulting text and model are given, respectively, in Figs. 23 and 24.

The work described above has brought to conclusions that support the using of this modeling methodology for identifying inconsistencies within or between standards, improving the quality and the value of standards. On the other hand, while learning OPM is easy, domain expertise and modeling experience were found to be critical in using its compact concept set and successful application. The limitations of the proposed method include also the level of effort required to develop insightful diagrams. Detailed analysis and evaluation results are discussed in [25].

IV. Conclusion

In this paper, we presented a MBSS methodology and Tesperanto—a supporting software environment for generating human-like authored natural English specification of the OPM model underlying the document. The motivation for our approach is the abundance of inconsistencies and unclear specifications in many technical documents, including international standards, underscoring the pressing need to produce documents that are more robust. Our approach treats a technical document as a product in its own right and hence applies SE methods and a holistic, lifecycle-encompassing treatment to the generation and maintenance of technical documents. The evaluation of MBSS in general and Tesperanto as part of it via examples form ISO standards and the Gestonurse case study has demonstrated the value of this paradigm as support tool and approach for authoring the systems' specifications in a way that they can be more aligned and complete with the authors goals than those generated by traditional text-centered idiosyncrasies.

Some drawbacks of the discussed methodology include a significant large amount of effort required to adopt existing platforms and tools in realistic cases and the need for domain-specific support.

In addition to the limitations discussed, we will further focus on extending this paper at both the technical and conceptual fronts. In terms of the technical improvements, we will extend the Tesperanto editor with new features and commands based on the feedback reported through user-based studies. At the conceptual front, we will include domain ontologies into the system. At the application level, systems such as Gestonurse will be adjusted to reflect and respond realistically to communications in the OR, thus reducing the number of miscommunications.

In parallel, together with the other participants in ISO TC/184 SC/5 WG/1, we have been developed OPM as an ISO publically available specification (19450), which was adopted recently and the ISO Draft International Standard for model-based standards authoring, which assumes OPM as the underlying conceptual modeling language and methodology. We expect Tesperanto with its implementation to play an important role in this emerging standard.

References

[1] A. Pyster, D. Olwell, A. Squires, and N. Hutchison, *A Guide to the Systems Engineering Body of Knowledge (SEBoK)*, INCOSE, San Diego, CA, USA, 2012.

[2] E. M. Blom *et al.*, "Analysis of verbal communication during teaching in the operating room and the potentials for surgical training," *Surg. Endosc. Other Interv. Tech.*, vol. 21, no. 9, pp. 1560–1566, 2007.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[3] A. Blekhman and D. Dori, "Tesperanto—A model-based system specification methodology and language," in *Proc. 23rd Annu. INCOSE Int. Symp.*, Philadelphia, PA, USA, 2013, pp. 139–153.

[4] I. Reinhartz-Berger and A. Sturm, "Utilizing domain models for application design and validation," *Inf. Softw. Technol.*, vol. 51, no. 8, pp. 1275–1289, 2009.

[5] *UN/CEFACT. Core Components Technical Specification Version 3.0 2nd Public Review*, United Nations Centre Trade Facilit. Electron. Bus., Geneva, Switzerland, 2007. [Online]. Available: http://www.unece.org/fileadmin/DAM/cefact/codesfortrade/CCTS/CCTS-Version3.pdf

[6] M. Fowler, *Analysis Patterns: Reusable Object Models*, vol. 3. Menlo Park, CA, USA: Addison Wesley, 1996, p. 361.

[7] C. Raistrick, "Model driven architecture with executable UML," vol. 1. Cambridge, U.K., Cambridge Univ. Press, 2004.

[8] C. Bock, "SysML and UML 2 support for activity modeling," *Syst. Eng.*, vol. 9, no. 2, pp. 160–186, 2006.

[9] P. Fritzson and V. Engelson, "Modelica—A unified object-oriented language for system modeling and simulation," in *ECOOP'98—Object-Oriented Programmig*. Berlin, Gremany: Springer, 1998.

[10] D. Dori, *Object-Process Methodology: A Holistic Systems Paradigm; With CD-ROM*. Berlin, Germany: Springer, 2002.

[11] A. Gamati, C. Brunette, R. Delamare, T. Gautier, and J.-P. Talpin, "A modeling paradigm for integrated modular avionics design," in *Proc. 32nd EUROMICRO Conf. Softw. Eng. Adv. Appl. (EUROMICRO)*, Cavtat, Croatia, 2006, pp. 134–143.

[12] A. Renick, S. Bolshchikov, J. Somekh, and D. Dori, "OPM simulation with computational interface to MATLAB and model-driven vivid animation," in *Proc. 6th Int. Conf. Syst. Eng. (INCOSE_IL)*, Herzeliya, Israel, 2011, pp. 193–198.

[13] *IEEE Recommended Practice for Software Requirements Specifications*, IEEE Standard 830, 1998.

[14] J. Kasser, "Object-oriented requirements engineering and management," in *Proc. Syst. Eng. Test Eval. Conf. (SETE)*, Canberra, ACT, Australia, 2003, pp. 1–15.

[15] Z. Liu, H. Jifeng, X. Li, and Y. Chen, "A relational model for formal object-oriented requirement analysis in UML," in *Formal Methods and Software Engineering*. Berlin, Germany: Springer, 2003.

[16] B. Bastani, "Process-oriented abstraction of the complex evolvable systems," *ACM SIGSOFT Softw. Eng. Notes*, vol. 33, no. 3, p. 1, 2008.

[17] *Enterprise Architect Requirements Management Documentation*. (Apr. 25, 2011). [Online]. Available: http://www.sparxsystems.com/downloads/whitepapers/Requirements_Management_in_Enterprise_Architect.pdf

[18] H. Liu and H. Lieberman, "Metafor: Visualizing stories as code," in *Proc. 10th Int. Conf. Intell. User Interf.*, San Diego, CA, USA, 2005, pp. 305–307.

[19] D. Harel, "Can programming be liberated, period?" *Computer*, vol. 41, no. 1, pp. 28–37, Jan. 2008.

[20] D. Harel, "From play-in scenarios to code: An achievable dream," *Computer*, vol. 34, no. 1, pp. 53–60, 2001.

[21] G. J. Fakas, "Automated generation of object summaries from relational databases: A novel keyword searching paradigm," in *Proc. Int. Conf. Data Eng.*, Cancun, Mexico, 2008, pp. 564–567.

[22] R. Barzilay and M. Lapata, "Modeling local coherence: An entity-based approach," *Comput. Linguist.*, vol. 34, no. 1, pp. 1–34, 2008.

[23] R. Barzilay and L. Lee, "Catching the drift: Probabilistic content models, with applications to generation and summarization," in *Proc. Human Lang. Technol. Conf. (HLT-NAACL)*, Boston, MA, USA, 2004, pp. 113–120.

[24] M. O'Donnell and A. Knott, "Optimising text quality in generation from relational databases," in *Proc. 1st Int. Conf. Nat. Lang. Gener. (INLG)*, Seattle, WA, USA, 2000, pp. 133–140.

[25] A. Blekhman, D. Dori, and R. Martin, "Model-based standards authoring," in *Proc. INCOSE Int. Symp*. Denver, CO, USA, Jun. 2011, pp. 650–659.

[26] P. Checkland, *Systems Thinking, Systems Practice: Includes a 30-Year Retrospective*. Chichester, U.K.: Wiley, 1999.

[27] A. Mandal and S. G. Deshmukh, "Vendor selection using interpretive structural modelling (ISM)," *Int. J. Oper. Prod. Manage.*, vol. 14, no. 6, pp. 52–59, 1994.

[28] A. Hari, J. Herscovitz, A. Zonnenshain, and M. P. Weiss, "Application of ICDM for the conceptual design of a new product," in *Proc. Int. Design Conf.*, Dubrovnik, Croatia, 2002, pp. 647–654.

[29] J. Firth-Cozens, "Why communication fails in the operating room," *Qual. Saf. Health Care*, vol. 13, no. 5, p. 327, Oct. 2004.

[30] L. Lingard *et al.*, "Communication failures in the operating room: An observational classification of recurrent types and effects," *Qual. Saf. Health Care*, vol. 13, no. 5, pp. 330–334, 2004.

[31] M. Jacob, Y.-T. Li, G. Akingba, and J. P. Wachs, "Gestonurse: A robotic surgical nurse for handling surgical instruments in the operating room," *J. Robot. Surg.*, vol. 6, no. 1, pp. 53–63, Nov. 2011.

[32] E. M. Blom *et al.*, "Analysis of verbal communication during teaching in the operating room and the potentials for surgical training," *Surg. Endosc.*, vol. 21, no. 9, pp. 1560–1566, Sep. 2007.

[33] S. S. Awad *et al.*, "Bridging the communication gap in the operating room with medical team training," *Amer. J. Surg.*, vol. 190, no. 5, pp. 770–774, 2005.

Alex Blekhman received the B.Sc. degree in computer engineering, the M.E. degree in electrical engineering, and the Ph.D. degree in information systems engineering from the Technion—Israel Institute of Technology, Haifa, Israel.

He has developed a generic formal procedure, an underlying ontology and language, and a complementary integrated software environment for the authoring and management of structured model-based documents during his Ph.D. thesis. His current research interests include systems engineering, project management, and information systems.

Juan P. Wachs received the M.Sc. and Ph.D. degrees in industrial engineering and management from the Ben-Gurion University of the Negev, Beersheba, Israel.

He is an Assistant Professor with the School of Industrial Engineering, Purdue University, West Lafayette, IN, USA, where he is the Director of the Intelligent Systems and Assistive Technologies Laboratory. He was a Post-Doctorate at the Naval Postgraduate School's MOVES Institute, Monterey, CA, USA, under a National Research Council Fellowship from the National Academies of Sciences. His current research interests include machine and computer vision, robotics, teleoperations, human–robot interaction, and assistive technologies. He has published in journals, including the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, the *Journal of American Medical Informatics*, and *Communications of the ACM*.

Dr. Wachs was the recipient of the Air Force Young Investigator Award in 2013. He is an Associate Editor of the IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS and the *Journal of Real-Time Image Processing*. He is a member of the Operation Research Society of Israel.

Dov Dori (SM'98) received the B.Sc. degree in industrial engineering and management from the Technion—Israel Institute of Technology, Haifa, Israel, the M.Sc. degree in operations research from Tel Aviv University, Tel Aviv, Israel, and the Ph.D. degree in computer science from the Weizmann Institute of Science, Rehovot, Israel, in 1975, 1981, and 1988, respectively.

He is a Harry Lebensfeld Chair in Industrial Engineering and the Head of Enterprise System Modeling Laboratory with the Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology. Since 2000, he has been an intermittently Visiting Professor at Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, USA. He has invented and developed object-process methodology, the new ISO 19450 publically available specification. He has mentored over 50 graduate students. His current research interests include model-based systems engineering, conceptual modeling of complex systems, systems architecture and design, software and systems engineering, and systems biology. He has authored over 300 publications, including journal and conference publications, books, and book chapters.

Prof. Dori was an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He is currently an Associate Editor of *Systems Engineering*. He was the Chair of nine international conferences or workshops. He is a fellow of the International Council on Systems Engineering and the International Association for Pattern Recognition, a member of Omega Alpha Association—International Honor Society for Systems Engineering, and a Senior Member of the ACM.