2010

Primitives: Design Guidelines and Architecture for BPMN Models

Michael zur Muehlen
Stevens Institute of Technology, mzurmuehlen@stevens.edu

Dennis E. Wisnosky
U.S. Department of Defense, dennis.wisnosky@osd.mil

James Kindrick
Jacobs Technology Inc., james.kindrick@jacobs.com

Primitives:
Design Guidelines and Architecture for BPMN Models

Michael zur Muehlen
Howe School of Technology Management
Stevens Institute of Technology
Hoboken, NJ 07030 USA
mzurmuehlen@stevens.edu

Dennis E. Wisnosky
Office of the Deputy Chief Management Officer
U.S. Department of Defense
Washington, DC 20301-3600
dennis.wisnosky@osd.mil

James Kindrick
Strategic Solutions Group
Jacobs Technology Inc.
Chantilly, VA 20151
james.kindrick@jacobs.com

Abstract

The Business Process Modeling Notation has emerged as a popular choice for representing processes among Business Analysts and Information Systems professionals. While the BPMN specification provides a rich syntax for the capture and representation of process models, it does not provide any guidance for the organization of the resulting models. As a consequence, large process libraries may become disorganized and hard to manage due to variability in abstraction levels, process interfaces, and activity descriptions. Based on the analysis of a process library in a US government agency we present a proposal for design guidelines and use our design guideline to qualitatively assess existing work on model quality guidance. To better organize models at different abstraction levels we propose a process architecture that allows for the systematic organization of BPMN models for different stakeholder concerns.

Keywords

BPMN, Design Guide, Process Architecture, Enterprise Architecture

INTRODUCTION

The documentation of organizational processes in formal models has been studied for several decades, and both process modeling techniques and commercial tools have progressed to allow for the cost-efficient capture of large-scale process models. Large-scale modeling projects can result in hundreds of distinct process models (Becker et al. 2003), leaving their organization and change management as an area that is rarely addressed in the academic literature (Raduescu et al. 2006). This lack of structure and organization of the process landscape leads to models that deviate in their degree of detail, abstraction, and choices on how to represent certain semantic content. Issues related to the connectivity of processes often arise during later stages of process improvement projects due to a lack of necessary interface standards (Becker et al. 2003). Heterogeneous representations of similar content leads to user difficulties in understanding models developed by others. These effects result in higher costs for the modeling project due to the additional time and effort required to understand and analyze the process models. In addition, a lack of representational standards may increase the complexity perceived by model users. Recent studies state that next to the usefulness of the models the reduction of their complexity is a major driver for their continual use (Davies et al. 2006).

A possible remedy for this situation is the use of modeling standards and architecture frameworks that govern how process models are designed, structured, (de-)composed, and linked. In this paper we discuss the development of design guidelines and architecture standards in the context of a large IS architecture project. In the next section we introduce the project context and discuss the issues that led to the development of a design guide, which is discussed in section three. Section four outlines the architecture guidance that was developed as part of the project. We conclude the paper with an outlook on future work.

A PROCESS ARCHITECTURE IN PRACTICAL USE

The Business Mission Area of the United States Department of Defense (DoD) is responsible for the business and financial infrastructure of the DoD. Due to the large number of systems that support logistics, financial transactions, contract management and other applications, data integration and compliance management are critical areas to ensure the continual functioning of the department's processes. In order to manage these integration issues better in 2005 the DoD began to specify the Business Enterprise Architecture (BEA) (http://www.bta.mil/products/bea.html) as a means to prioritize improvement efforts, understand the impact of systems modernization projects on other enterprise systems, and to create an end-to-end representation of the core processes of the Business Mission Area.

The BEA is organized in six core areas that cover financial management, supplier management, materials management, facility management, human resources management and procurement. Each of these areas is documented through a series of business processes modeled in BPMN. In addition, business rules, data standards, system interfaces, and other models are maintained in views that conform to the DoD Architecture Framework (DoDAF 2.0) (U.S. Department of Defense, 2009). Overall the BEA contains several hundred models and is updated on a continual basis. At the time of this writing version 7.0 represents the most current release and can be accessed at www.bta.mil/products/bea_7_0/index.htm.

From a process management perspective two types of models in the BEA are of particular interest: The end-to-end business process models (DoDAF view OV-6c), and the functional decomposition of activities (DoDAF view OV-5a). While the process models are created in BPMN, the models describing functional decomposition and the activity inputs and outputs were created using IDEF0. Other views are described in tables, entity-relationship diagrams and proprietary diagram types.

Our analysis of the process models uncovered that over time modelers created various modifications to BPMN at the behest of their stakeholders, leading to technically incorrect BPMN models. A typical example was the use of combined start events. If the receipt of any of three documents could trigger a process, modelers often used a complex start event and attached message start events for each trigger to the complex start event, in violation of the BPMN syntax rules. Another example was the use of rule events attached to process activities to indicate that an activity was based on the use of business rules. Properly interpreted, these attached events would have terminated their host activities in case the rules fired, but that was not the semantics intended by the modelers.

In addition, the available BPMN constructs were being used inconsistently. For example, while some modelers used gateways to denote splits and joins in processes, others used conditional sequence flow to create splits and multiple incoming sequence flows into activities to create joins. As a result the same semantic content looked different depending on who created the model. This problem was exacerbated by the frequent turnover of process modelers, as each generation of modelers introduced their personal style into the resulting diagrams. In response to this discovery, the CTO and Chief Architect of the Business Mission Area commissioned the development of a modeling standard that was designed to enforce a common representation of processes across subject matter areas and modelers. The insights into design guideline development gained over the course of this project, called Primitives, are described in section 3. The first release of the design guidelines can be accessed at the following URL: http://www.bta.mil/products/BEA_7_0/index.htm under the heading "Federation".

PROCESS MODELING GUIDELINES

The academic literature contains some recommendations for the design of high-quality process models. Becker et al. proposed the Guidelines of Modeling (GoM), which consist of six criteria that can be used to determine the quality of a model. These criteria are semantic and syntactical correctness, relevance of content, economic viability of model creation, clarity of representation, comparability to related content, and systematic design (Becker et al. 2000). Krogstie et al. (2006) describe an extended version of the semiotic quality framework SEQUAL, which organizes modeling guidelines based on different quality levels (e.g. syntactic, semantic, physical or empirical quality).

Both of these approaches share that they are not rooted in a particular modeling method, but can be applied to different kinds of conceptual models, for example data models as well as process models. Mendling et al. (2010) criticized these approaches as theoretic and not useful for inexperienced modelers. In response, they proposed a series of seven design heuristics (7PMG) focused on the design of Event-driven Process Chains. These guidelines were derived through a series of student experiments and surveys at participating universities as well as the analysis of the reference models that ship with the SAP ERP solution. These guidelines are: (1) minimize the number of model elements, (2) minimize routing paths, (3) use one start and end event, (4) model as structured as

possible, (5) avoid OR routing elements, (6) use verb-object activity labels, and (7) decompose the model if it has more than 50 elements.

Primitives – A Practical Set of Modeling Guidelines

The Primitives design guide consist of three parts: A subset of the BPMN 2.0 symbols (the graphical part), a collection of design patterns for the use of these symbols, and design rules that cover design aspects not covered by the patterns or the symbol set. The development of our BPMN modeling guidelines began with an analysis of the modeling habits present in the existing BEA models. We identified the most commonly used BPMN symbols, and combined this information with a previous study of the application of BPMN as a baseline for the development of a BPMN subset (zur Muehlen & Recker 2008). We stripped out the least used BPMN constructs and limited the use of others where equivalent representation could be achieved by other means. For example, we outlawed the use of conditional sequence flow and mandated the use of gateways for splits and joins. To determine if the resulting subset of BPMN symbols was sufficient for use within the DoD, we applied the constrained BPMN vocabulary in the design of processes outside of the BEA, namely the Joint Close Air Support (JCAS) process, and several governance processes. Figure 1 shows the selection of BPMN symbols that constitute the primitives subset.

To assist modelers in the construction of primitives-compliant models we paid particular attention to syntactical errors modelers had made in the BEA processes. We created reference patterns that would represent the intended semantics correctly, and added them to a library of patterns that was complemented by design patterns from the Workflow Patterns library (van der Aalst et al. 2003). While testing the primitives in the JCAS process we identified additional patterns that covered approvals, collaborative decision-making and similar scenarios that were added to the modeling library.

Applying the 7PMG and the GoM in Practice

In the context of the Primitives project we evaluated the heuristics developed by Mendling et al. (2010) with the intention to add them to our set of design guidelines. However, while the seven guidelines are intuitively accessible, we found some of them easier to implement than others. For example, while minimizing the model elements and control flow paths is an intuitive approach to limiting the complexity of a model, we found that the amount of model content was driven more by stakeholder concerns than modeler discretion. On the other hand, we did prescre a verb-object style for activity labels, complemented with guidance to avoid terms like “manage”, “perform”, or “process”. Additionally, developing the Primitives allowed us to assess the feasibility of Becker et al.’s Guidelines of Modeling. Using our experiences as an evaluation guide for the GoM, we came to several conclusions:

- Of the six guidelines proposed by Becker et al., *syntactical correctness* is the easiest to implement, as it simply states that any chosen modeling method has to be applied correctly. By employing syntax checks in modeling software and manual reviews of models it is feasible to enforce the design of technically correct models. This result can be accelerated if the underlying modeling method is reduced in complexity.
- *Semantic correctness* can only be established in the context of a particular domain and model, making it very difficult to provide modelers with generic actionable guidance in this regard. Only a subject matter expert that is conversant in a chosen modeling method can determine if a given model represents the subject matter correctly.
- *Relevance* enforces a minimalist principle on process models. Models are supposed to contain no more information than necessary, but also no less information than required by their purpose. Since the purpose for the BEA models is well documented (compliance management) it was straightforward to establish which content would be required to meet the relevancy criterion. However, in practical use the question of what model elements are deemed relevant turned out to be a contentious issue. BPMN consists of a visual notation and an underlying metamodel that includes attributes for the visual modeling elements. This allows for the representation of information at both the visual and textual level. Since the main purpose of the BEA is compliance management, data input and output relationships are an important aspect of the BPMN models. Several stakeholders insisted that both mandatory and optional data elements were displayed in the BPMN models. Since BPMN does not provide a notation for mandatory and optional data associations each data element in the resulting models looks to be of equal importance. This issue could be addressed at the attribute level, as InputSets and OutputSets of BPMN tasks can have mandatory and optional elements. However, BEA stakeholders frequently interact only with the graphical representation and have no insight into which attributes are maintained in a given model.

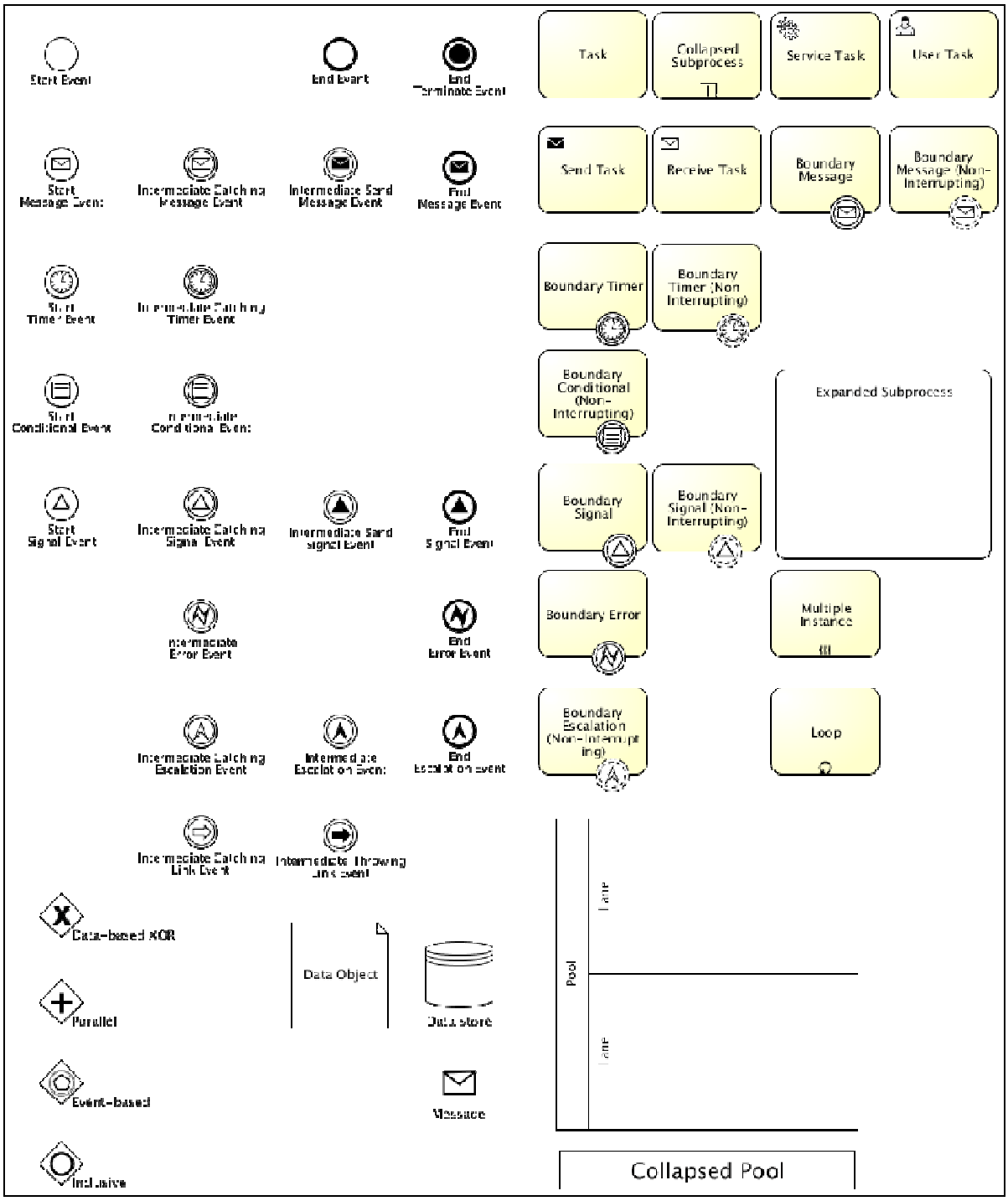

Figure 1: BPMN Primitives Symbol Subset

- *Economical design* can be supported by using predefined model building blocks, so process designers don't have to create every model from scratch. The use of patterns in software architecture is well established (see e.g. Zdun et al. (2007)), and the use of reference models has a longstanding tradition in process modeling (see e.g. Fettke et al. (2005)). We used the Workflow Patterns library as a starting point to define elementary building blocks (van der Aalst et al. 2003). When we applied the design guide to both the BEA and several other process modeling projects we discovered additional patterns that incorporated domain semantics, for example a document release, a collaborative decision-making step, or messaging patterns. The design and management of this emerging pattern library is an ongoing topic within the project and its description is beyond the scope of the current paper.
- *Clarity*, *comparability*, and *systematic design* are optional quality attributes according to the Guidelines of Modeling. Some of Mendling et al.'s heuristics address the clarity aspect of process modeling by targeting the cognitive effort necessary to understand a given model, and we adopted these heuristics were

appropriate. For example, we ask modelers to use verb-object labels in activities and require models to have one defined start point. However, we permit multiple exit points due to the presence of positive and negative process outcomes that are handled differently. We found that once rules for syntactical correctness are established, enforcing clarity of representation becomes a smaller problem to manage.

- *Comparability* relates to the traceability of content between different models, in particular when different techniques were employed. In our case the alignment between the activity tree (OV-5a) and the process models (OV-6c) presented a big issue that we are trying to solve through defined levels of abstraction as outlined in section 4 of this paper.
- *Systematic Design* requires process models to follow a logical structure. In our case, this structure was provided by another architecture: The DoD has defined Joint Capability Areas (JCAs) that describes at the highest level what desired effects it wants to be able to create. The ability to create a trace from a process to its objectives is a highly desirable feature to ensure the alignment between operations and strategy. For this reason the next revision of the BEA is based on an analysis of capabilities supplied by each end-to-end process. These capabilities will then be compared to the JCAs in order to identify coverage areas and potential gaps.

Design Guidelines and Tool Support

Of the design guidelines developed in the context of the Primitives project the tailoring of the BPMN vocabulary turned out to be the biggest step towards ensuring a consistent representation of processes created by different modelers. However, the design of such a subset is only feasible if the modeling tool supports it. While many enterprise architecture tools allow administrators to create custom method subsets (for instance ARIS, Mega, System Architect and others), the interchange of the resulting models is hampered by the absence of formal interoperability tests and compliance standards. For example, BPMN up to version 1.2 did not specify any means by which a vendor could claim conformance with the standard unless all BPMN elements were supported by the tool. While this approach is feasible for diagramming tools, execution tools that instantiate and automate BPMN processes (i.e., BPMS) often support only a subset of the BPMN syntax. This is often due to the limitations of the underlying execution infrastructure. For example, a BPMS typically assumes full control over all elements of the process model that is created in the BPMS design environment. In BPMN this means that all process elements reside within a single pool, as the pool element represents one coherent unit of control. As a consequence, many BPMS do not support multiple pools, and thus do not support message flow symbols, as message flow can only occur between elements in different pools.

The BPMN 2.0 specification contains three conformance subclasses that describe subsets of the BPMN vocabulary for different uses, Descriptive, Analytical and Common Executable (OMG 2010). The choice of these subclasses is based on work by Silver (2009), contributions from Robert Shapiro and the Primitives project. The first subset, descriptive, contains a very limited vocabulary in order to facilitate model understanding by a broad group of stakeholders. The second subset, analytical, adds more refined modeling elements such as different event types and exceptions. The third level, minimum executable, focuses mainly on the attributes of BPMN elements that need to be maintained to support the execution of a BPMN process. The subset defined as part of the Primitives project provided key input for the design of the analytical conformance subclass. Using the different subclasses vendors can formally claim conformance to a defined subset of the BPMN vocabulary. This in turn will lead to clearer BPMN serializations based on these subsets, which should make it easier to exchange models between different tools.

ARCHITECTURE GUIDANCE

While the BPMN design guidance was developed to increase the quality of the process models in the BEA, it did not address the alignment of these models with other architecture views, most importantly the activity hierarchy described in the DoDAF view OV-5a. A closer inspection of the BEA activity decomposition model uncovered several issues in the design of the activity hierarchy. One issue was the inconsistent use of levels across the hierarchy. Some parts of the activity tree were three levels deep, others had six levels of decomposition. As a result, deriving a consistent process that linked different branches of the tree was a difficult task. Another issue was inconsistently applied methods of decomposition. For example, some activities were decomposed using temporal criteria; others were decomposed based on the object manipulated by the activity, leading to inconsistent activity labels. Finally, the activity decomposition was not aligned with the detailed process. While some activities in the decomposition model were operational and could map into the BPMN models (e.g. accept goods and services), others represented support activities that were performed on a continual basis and had no counterparts in the end-to-end processes (e.g. manage union relations program). As a result, the architecture was difficult to integrate across the different viewpoints.

Some of these issues arose because different functional units were responsible for different branches of the activity hierarchy, others were due to the fact that the activity tree and the process models were created for different stakeholder groups. These issues persisted, even though a detailed architecture development guide was in use for the design of BEA models. The resulting question was: Could a process architecture with formally defined levels provide a better structure for designing both the BPMN models and the activity tree?

Enterprise Architecture

Enterprise Architecture deals with the organization of the enterprise's resources and brings together business-oriented models (such as process and organizational models) with technical models (such as system interface descriptions) (Davis & Brabänder 2007). The resulting artifact of Enterprise Architecture is a set of descriptive representations that are relevant for the management of an enterprise and can be maintained over the period of its useful life (Zachman 1999). The task of an enterprise architect is to provide a common view of the primary resources of any enterprise (people, processes and technology) and to demonstrate their integration to support the strategic goals of the organization (Anaya & Ortiz 2005). Examples of architecture frameworks are Kruchten (1995); Hilliard (2000) and Zachman (1999).

The process view is a key element in many architecture frameworks, as it describes the behavioral properties of a system. Consider e.g. Koliadis et al. (2008) who compare different Enterprise Business Process Architectures. The models that make up the BEA follow the perspectives of the DoD Architecture Framework 2.0 (U.S. Department of Defense 2009). However, this framework does not provide guidance for the design or organization of models within the different perspectives. The growing number of process models in the BEA needed to be organized along a process architecture.

Abstraction and Generalization

The purpose of architecture is to provide structured means for the management of complexity. A common measure to manage this complexity is the use of abstraction. Abstraction is a process in which designers represent only those details they deem relevant in their models, while leaving out details that are perceived as irrelevant. The relevance of content depends on the modeling purpose and interests of model users and maps directly to the GoM principle of relevancy. Two major forms of abstraction are aggregation and generalization (Smith & Smith 1977):

- Aggregation describes the relationship between objects as a higher-level object that is composed of the lower-level parts. In the context of process management aggregation refers to the representation of horizontally linked processes through a higher-level element, such as the subprocess symbol in BPMN. The creation of composite process steps from multiple underlying steps can be described as composition (and the inverse as decomposition).
- Generalization describes a set of related objects as a higher-level object by ignoring the differences, e.g. different attribute values. In the context of process management generalization is implicit in the design of a model that captures the commonalities among different process instances.

In short, aggregation is a lossless mechanism to create higher (and lower) levels of abstraction, while generalization/specialization typically involves the removal (or addition) of information. Whereas aggregation and generalization originated in the field of data management as a way to manage diverse data sets in a bottom-up fashion, many process management projects employ a top-down notion of process development. Consequently, a key element of many process architectures is the notion of model composition and decomposition in order to manage the complexity at each architecture level. For the purposes of our project we reviewed several examples of process architectures:

- Davis reports on the use of a six-level process architecture at British Telecom, where the top three levels focus exclusively on process outcomes and measurements, while the bottom three levels focus on delivery mechanisms and execution support (Davis 2006). For each level a design guide has been developed that governs the available modeling elements at each level.
- Similarly, the DeTeImmobilien enterprise modeling project used up to six different levels of process hierarchy to manage the large number of process models (Becker et al. 2003), although no formal separation criteria for process decomposition were used beyond the third level.
- Sony pictures used three different levels of abstraction to redesign their merchandise management operations (Hunter, 2007), based on recommendations by Sharp & McDermott (Sharp & McDermott 2008): A handoff level that focused exclusively on work crossing organizational boundaries, a milestone

level that focused on significant decision making points in the process, and a task level that focused on the procedural logic necessary to complete each process step.

- The IDEF0 methodology (see e.g. NIST (1993)) contains design guidance that limits the number of process elements to 4-6 per page and requires a decomposition if the process requires further elements. IDEF0 does not impose any constraints on the levels of composition/decomposition that can be used.

Some architecture frameworks provide guidance for the levels of decomposition an architecture should contain.

- The ARIS framework divides each view into three levels: The conceptual model, requirements engineering, and implementation specification. This distinction is derived from the architecture levels found in data management.
- The Zachman framework distinguishes between six levels of abstraction using building architecture as an analogy: Scope (ballpark view), business model (owner's view), IS model (designer's view), technology model (builder's view), detailed description (out-of-context view), and the actual system (without a formal description).
- The Object Management Group distinguishes between a Platform Independent Model (i.e. a model that does not contain features that necessitate a particular implementation) and a Platform Specific Model (i.e. a model that is tightly coupled to a particular implementation technique).
- The DoDAF framework distinguishes between two main levels: An operational view (for requirements engineering), and a systems view that is complemented by a parallel services view (for systems engineering). DoDAF contains various other views that cover capability analysis, project management, general project description, technical standards, information models and other aspects that are not directly applicable to our process focus.

Model Decomposition versus Model Enrichment

Within both the operational view and the services/systems view DoDAF specifies a perspective for process models: Operations View models (OV-6c) at the requirements engineering level, and Systems and Service View models (SV-10c/SVcV-10c) at the systems engineering level. The presence of these different levels is driven by the needs of different user communities: Models at the operations view level are typically used to capture the business requirements for a system, while models at the systems/services view level are used to create implementable specifications. The amount of information collected at each level depends on the intended use. For example, a BPMN model created at the OV-6c level often can serve its purpose through the graphical process representation alone, while an executable BPMN model at the SV-10c level is only executable if the modeler maintains attributes of BPMN elements that are not visible at the graphical level. The transition from OV-6c to SV-10c is thus not necessarily a decomposition of a higher-level model to a lower-level model. Instead, an existing model is enriched with additional information, making it a generalization/specialization relationship.

The consequence of this relationship is that the design guidelines for models at different levels of abstraction can vary, as can the vocabulary recommended to the modelers. In the case of Primitives we chose to keep the same BPMN symbol set at both the OV-6c and SV-10c levels, but to require fewer attributes for the model elements at the OV-6c level. However, this measure by itself does not guarantee a consistent composition/decomposition strategy for the models within each level. For this purpose, we reviewed some of the design guidance mentioned in section 4.2. The outcome was a recommended set of modeling levels for analytical purposes, i.e. the OV-6c level. It is our expectation that the modelers that design for implementation will begin at the lowest level of our hierarchy and refine their models from there. The resulting recommendations are described in the next section.

Architecture Levels for BPMN Modelers

We applied our design guidance in several projects outside of the BEA in order to refine our choice of modeling symbols and to better understand the different levels of abstraction needed in the DoD process architecture. One of these projects was the creation of BPMN models for a highly collaborative process that had been documented exclusively in textual form. The intended use of the process model was the identification of communication capabilities and requirements among the different process participants. The modelers worked in an online tool together with subject matter experts that had no previous experience with BPMN. The resulting models emerged in a "middle-out" fashion: First, a relatively detailed model of successful process execution was created and validated. Then, additional decisions and negative pathways were added to the model. When the complexity of the model suggested that certain process steps should be detailed separately it became necessary to create a high-level model that would define the horizontal interfaces between process models at a lower level. This milestone model contained no swimlanes and consisted of just six subprocesses, but it led to the refactoring of the initial model, as

the start and endpoints were better defined. Finally, one phase of the milestone model was chosen for further inspection and three additional levels of process models were created – one focusing on organizational interfaces, one as a segment of the initial model, and a fourth as a detailed version of a collaboration pattern within the third level. For the purposes of requirements identification we found the resulting four levels to be adequate. A systems engineer could take the lowest level model and treat it as the top-level model for the creation of technical specifications and implementable designs. The four architecture levels for analysis models are:

Level 1: Milestones: The milestone level mainly deals with the question of what the process is designed to accomplish, and serves to delineate discrete phases of the process that end in milestones. These milestones could be natural exit points for processing, or represent hand-offs between different units of responsibility. A milestone diagram can be represented as an end-to-end process in BPMN but also e.g. as a value chain diagram. Within this level no further decomposition should occur, but the collapsed subprocess symbol can be used to create linkages to other BPMN diagrams at the lower levels. The objective of this level is to provide an overview for all stakeholders of the process landscape for their orientation and motivation. The milestone level should provide a static model that only changes when the nature of the process changes. Data is not explicitly described at this level, but the overall inputs and outputs of the process can be documented through the use of message start and end events. Figure 2 shows a process for the generation of Business Intelligence reports at the milestone level.

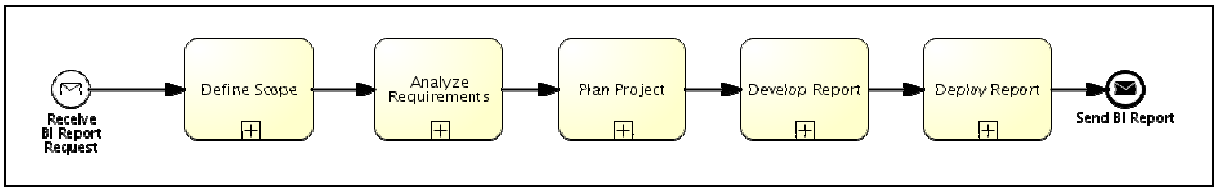

Figure 2: Milestone-level Process Description

Level 2: Handoffs: The handoff level introduces organizational units to the process model and focuses on boundaries of responsibility. It shows at what point of the process which role has control over the process (or parts thereof). The heuristic we applied was that there should be as little sequence flow within a swimlane (or pool) as possible. The activities within individual swimlanes (or pools) are treated as collapsed subprocesses. The handoff level can involve external business partners or services that are contacted via messaging.

The purpose of the handoff level is to identify possible communication bottlenecks in the process and frequent failure points due to communication errors. No data objects within lanes are used at this level, unless they related to the information flow across organizational boundaries. In this case message symbols and data objects can be used detail the information exchange between parties (what is sent versus what is expected). Figure 4 shows the same process as above at the handoff level.

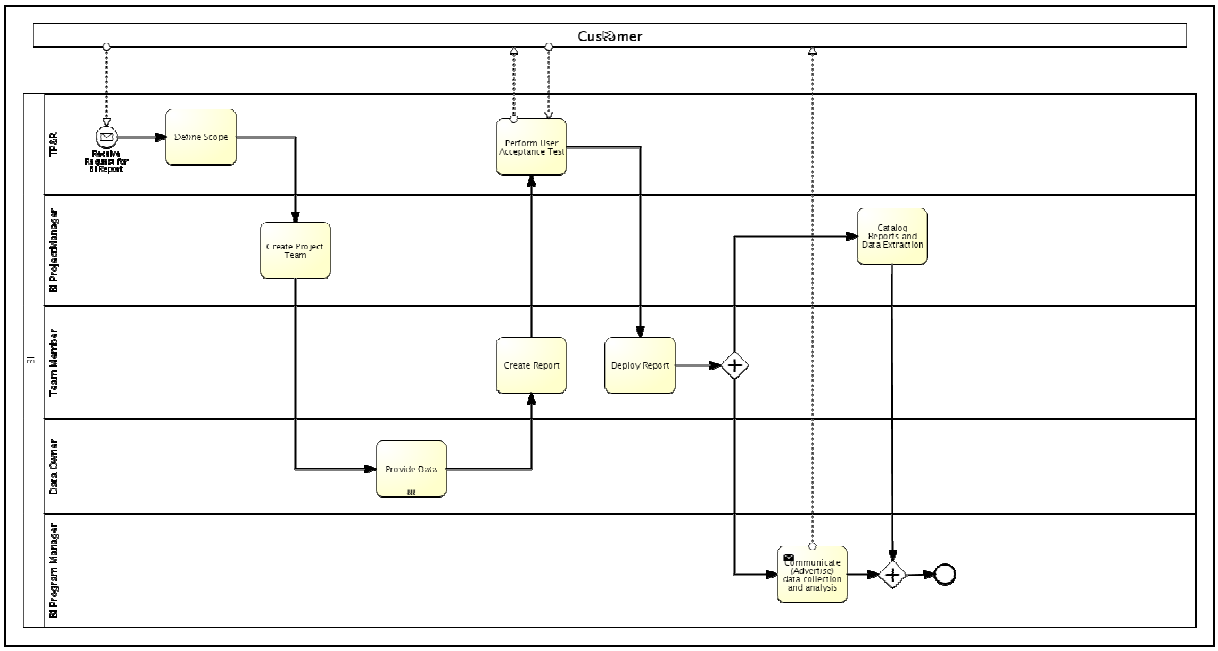

Figure 3: Handoff-Level BPMN Process

Level 3: Decisions: The decision level focuses on those steps in the process that can affect the positive (or negative) outcome of a case. It details the information required to perform decision-making activities and

describes their outcome. The resulting BPMN models may contain multiple pools and lanes, and sequence flow within lanes is permissible as long as it relates to decision-making. Any steps that are not related to decision-making (e.g. obtaining or recording information) are integrated as much as possible. The purpose of the decisions level is to support the identification of decision responsibilities (who decides?), the basis for these decisions (what data is required?) and the possible outcomes (what are the choices?). The focus of this level is typically on human decision-making, that is, technical choices such as the handling of unsuccessful service invocations or quality of service decisions such as timeouts are not addressed at this level.

Level 4: Procedures: The procedure level details the individual steps a role or participant has to perform in order to complete a process instance. This level introduces the mechanisms used for the completion of tasks, i.e. the BPMN task types (manual, service, user etc.) can be used to indicate how a particular process step is performed. Models at the procedure level can become the top-level processes for a BPMS implementation, as they take the perspective of individual units of control. Quality of service criteria such as timeouts and error handling are added to the process models at the procedure level as well.

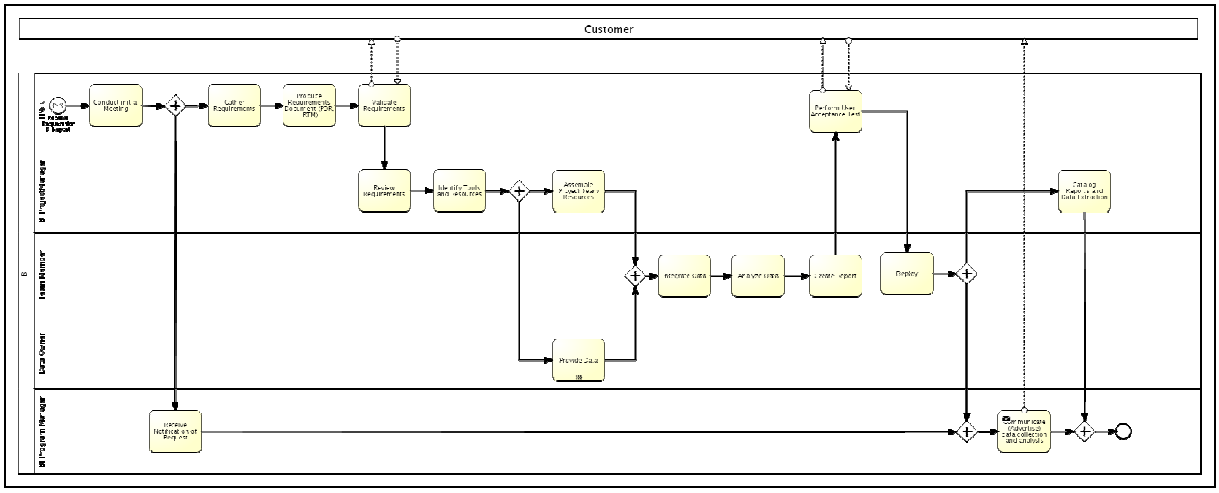

Figure 4: Procedure-level BPMN Diagram

SUMMARY AND OUTLOOK

In this paper we have outline the design and architectural guidance developed as part of an industry project within the US Department of Defense. The design guidelines were derived from the Guidelines of Modeling, empirical studies of BPMN in practice and the Workflow Patterns literature. They were refined and evaluated in several case studies and have become part of the design guidance for process modelers in the context of the DoD Architecture Framework. As part of the architecture guidance we have inductively developed a four-level framework for the organization of BPMN models according to different stakeholder concerns. We have validated this framework in different modeling scenarios within the Department of Defense.

We believe that the design guidance developed in this project can easily transfer to other organizations that need to manage the quality of BPMN models in large-scale modeling projects. As for the choice of a level architecture, our experience indicates that the modeling purpose has a significant influence on which levels are appropriate in a given situation. However, BPMN modelers need to distinguish between different architecture levels that address the concerns of different stakeholders, and different levels of abstraction within one architecture level that are used to manage the complexity of the resulting models. The large variety of level recommendations in the academic and popular literature suggest that there may not be a universal answer to the question how many levels a process architecture should have.

We are currently investigating the linkages between the process models in the BEA and other architecture perspectives such as capability models, business rule models, and system interface descriptions. If the architecture levels and design guidance developed for BPMN can transfer to other perspectives we may be able to lower the effort required to maintain large-scale enterprise architectures.

REFERENCES

Anaya, V., & Ortiz, A. (2005). *How enterprise architectures can support integration*. Proceedings of the first international workshop on Interoperability of heterogeneous information systems, Bremen, Germany.

Becker, J., Kugeler, M., & Rosemann, M. (2003). *Process management: a guide for the design of business processes.* New York et al.: Springer.

Becker, J., Rosemann, M., & Von Uthmann, C. (2000). Guidelines of business process modeling. In W. M. P. van der Aalst et al. (Ed.), *Business Process Management: Models, Techniques, and Empirical Studies* (pp. 30-49). Berlin et al.: Springer.

Davies, I., Green, P., Rosemann, M., Indulska, M., & Gallo, S. (2006). How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, *58*(3), 358-380.

Davis, R. (2006). *BPM in British Telecom - Practical Experiences*. Proceedings from Proceedings of the Australasian ProcessDays 2006, Sydney, Australia.

Davis, R., & Brabänder, E. (2007). *ARIS design platform: getting started with BPM.* New York: Springer.

Fettke, P., Loos, P., & Zwicker, J. (2005). *Business process reference models: Survey and classification*. Proceedings from Proceedings of the Business Process Management Workshops.

Hilliard, R. (2000). IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems.

Hunter, G. (2007). *Case Study: How Business Process Mapping Saved an IT Project*. Proceedings from Proceedings of the 2007 IIR BPM Conference, San Diego, CA.

Koliadis, G., Ghose, A. K., & Padmanabhuni, S. (2008). *Towards an Enterprise Business Process Architecture Standard*. Proceedings from IEEE Congress on Services-Part I, 2008, Waikiki, HI.

Krogstie, J., Sindre, G., & Jørgensen, H. (2006). Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, *15*(1), 91-102.

Kruchten, P. (1995). Architectural Blueprints—The "4+1" View model of software architecture. *IEEE Software*, *12*(6), 42-50.

Mendling, J., Reijers, H. A., & van der Aalst, W. M. P. (2010). Seven process modeling guidelines (7pmg). *Information and Software Technology*, *52*(2), 127-136.

NIST. (1993). Integrated Definition for Function Modeling (IDEF0). *Draft FIPS 183*. National Institute for Standards and Technologies, Gaithersburg, MD.

OMG. (2010). Business Process Model and Notation (BPMN 2.0). Document Number bma/2010-06-03. Object Management Group, Framingham, Mass.

Raduescu, C., Tan, H.-M., Jayaganesh, M., Bandara, W., zur Muehlen, M., & Lippe, S. (2006). *A Framework of Issues in Large Process Modeling Projects*. Proceedings from Proceedings of the 14th European Conference on Information Systems (ECIS 2006), Göteborg, Sweden.

Sharp, A., & McDermott, P. (2008). *Workflow Modeling: Tools for Process Improvement and Applications Development.* Artech House Publishers.

Silver, B. (2009). *BPMN Method & Style.* Altos, CA: Cody-Cassidy Press.

Smith, J. M., & Smith, D. C. P. (1977). Database abstractions: aggregation and generalization. *ACM Transactions on Database Systems (TODS)*, *2*(2), 105-133.

U.S. Department of Defense. (2009). DoD Architecture Framework Version 2.0. Volume 1: Introduction, Overview and Concepts. Manager's Guide. *Version 2.0*.

van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, *14*(1), 5-51.

Zachman, J.A. (1999). A framework for information systems architecture. *IBM Systems Journal*, *38*(2/3), 454-470.

Zdun, U., Hentrich, C., & Dustdar, S. (2007). Modeling Process-Driven and Service-Oriented Architectures Using Patterns and Pattern Primitives. *ACM Transactions on the Web*, *1*(3).

zur Muehlen, M., & Recker, J. (2008). How much language is enough? Theoretical and Practical Use of the Business Process Management Notation. *Conference on Advanced Information Systems Engineering (CAiSE '08)*.

COPYRIGHT