Measuring Method Complexity: UML versus BPMN

Jan Recker
Information Systems Program
Queensland University of Technology
j.recker@qut.edu.au

Michael zur Muehlen
Howe School of Technology Management
Stevens Institute of Technology
Michael.zurMuehlen@stevens.edu

Keng Siau
College of Business Administration
University of Nebraska-Lincoln
ksiau@unlnotes.unl.edu

John Erickson
College of Business Administration
University of Nebraska-Omaha
johnerickson@mail.unomaha.edu

Marta Indulska
UQ Business School
The University of Queensland
m.indulska@business.uq.edu.au

ABSTRACT

Graphical models are used to depict relevant aspects of real-world domains intended to be supported by an information system. Various approaches for modeling exist and approaches such as object-oriented and process-oriented modeling methods are in widespread use. These modeling methods differ in their expressive power as well as in their complexity of use, thereby leading to an important investment decision for organizations seeking to conduct modeling projects. In this paper, we used an established approach for evaluating the complexity of conceptual modeling methods and compared two important industry standards for modeling, Unified Modeling Language and Business Process Modeling Notation, based on their complexity. Our research finds that BPMN has very high levels of complexity when contrasted with UML.

Keywords (Required)

Process modeling, object-oriented modeling, complexity, UML, BPMN.

INTRODUCTION

In the current unsettled economic environment, businesses and organizations are increasingly looking for new ways to trim costs. Software development and maintenance costs represent a significant percentage of capital spending by business, some estimates rate it as high as fifty percent (Laudon and Laudon 2005). Many different means are often proposed to cut operational and maintenance costs, but cutting development cost is not usually a large topic of discussion. However, if systems can be better designed up front, the maintenance and operational costs can be favorably impacted or reduced.

One way to better develop systems is to do a more thorough job of systems analysis and design. A better understanding of the business and its context can often help point the way to better information systems. One of the tools commonly used to capture and portray an existing or new system is conceptual modeling (Wand and Weber 2002). Conceptual modeling can provide a bridge between the technical developer's side of the project and the more user oriented business side of the project, and overall reduces the likelihood of design errors at an early stage of systems development. While it is not a necessity for users to understand the complete technical details of any given project, it is also not desirable to convey information about the meta-model to the developers or the business users. However, if developers are to effectively use the modeling methods in their efforts to make the resulting system models understandable by all parties, the modeling methods, by which models of existing or new systems are designed, should be relatively easy to use and the underlying meta-models complete enough to create accurate and useful models. If the modeling method proves too complex and/or difficult to use, it will probably not be used at all, or used incorrectly. Similarly, if a modeling method is not complete enough to allow a developer to articulate all relevant details of the system, the modeling method will not be useful and the resulting models will be incomplete and inaccurate.

Over time, modeling methods have proliferated widely (Siau and Wang 2007, Siau and Rossi 2009). This trend has continued into areas of modern modeling such as object-oriented analysis (Coad and Yourdon 1990), enterprise systems interoperability (Green et al. 2005), workflow design (Jablonski 1995), and business process modeling (Recker et al. 2009). In light of such proliferation, an increased need for theory and research has arisen to give some guidance and insight to practitioners to assist them in the comparison, evaluation, and use of available methods (Moody 2005).

One important aspect in the use of a modeling method is its complexity. Method complexity is a significant factor because it can affect the learnability, ease of use, and overall usage of a method, potentially enhancing longevity of the method and overall success of the modeling projects (Rossi and Brinkkemper 1996). The ability to measure the complexity of available methods can help developers and users alike by guiding the method development and selection processes. Even though the measure of complexity may not be perfectly accurate, a rough estimate of a method complexity is better than no information. Besides, the accumulation of research in the area will help to build and advance the field.

This research does not propose to judge whether complexity is inherently good or bad, nor does it attempt to pass judgment on how modeling methods are or should be used. Rather, we considered two standard modeling methods, UML and BPMN, and compared their relative complexities. We based our comparison on two separate studies, that of the complexity of UML (Siau and Cao 2001) and that of BPMN (Indulska et al. 2009), both of which use the Rossi and Brinkkemper (1996) complexity metrics. We acknowledge that complexity is but one of the set of important metrics of a modeling method (e.g., communication ability, preciseness), and that there are many tradeoffs between such metrics, e.g., between complexity and the ability to communicate, see for example Gemino and Wand (2005). Yet, complexity is recognized as an appropriate measure for a method's usefulness (Siau and Cao 2001), which justifies our interest in this metric. Our research allows practitioners to make informed decisions about a modeling method adoption decision, which is significant because the decision for a particular modeling method is associated with related investments in tool purchases, training, conventions, and methodologies (Iivari 1996).

In our analysis of the complexity of modeling methods, we focused on the two methods, UML and BPMN, because both methods have achieved widespread adoption in industry practice (Dobing and Parsons 2006; Recker 2009), and both are official industry standards of the Object Management Group (www.omg.org). The paper is organized as follows: First, a brief background of conceptual modeling using the selected standard methods, UML and BPMN, is presented. This is followed by a brief introduction to the two standards, UML and BPMN. A discussion of evaluation techniques for modeling methods followed. Then the research approach used in this paper is presented. A description of findings and results is followed by a discussion of those results. Finally, we present the conclusions and implications of the paper and outline our future research on this topic.

BACKGROUND

Conceptual Modeling with UML and BPMN

Graphical models are a conducive way of articulating knowledge about a domain and can help identifying errors at early stages of IS development (Moody and Shanks 2003). Indeed, conceptual modeling has become an essential cornerstone of many information systems analysis and design (ISAD) methodologies. Traditionally, modeling for ISAD has focused on data descriptions (e.g., Chen 1976). Since the 1990s, however, a renewed interest in other forms of conceptual modeling has occurred due to a number of developments. These developments such as the emergence of the object-oriented approach to software engineering (Ambler 2004), the emergence of process-oriented approaches to managing organizations and systems (Dumas et al. 2005), the design and use of workflow-based information systems, and the challenges arising from diffused, distributed, and large user cohorts and inter-organizational system interactions, have all contributed to the increasing interest.

Conceptual models (scripts) of IS domains are constructed using a modeling method (a procedure for constructing models, such as the UML method for object-oriented systems analysis and design). Given that typically various aspects of systems have to be modeled (e.g., structural, behavioral, interactional), a modeling method usually embodies a range of modeling grammars, each of which consists of a set of graphical constructs and rules to combine those constructs and focuses on one particular modeling aspect. UML 2.0 (Ambler 2004), for instance, comprises thirteen different modeling grammars, including use class diagrams, class diagrams, sequence diagrams, timing diagrams, etc.

In deciding how to model a real-world domain intended to be supported by an information system, the decision of the type of method to be used for conceptual modeling is an important consideration. The organization of modeling constructs in a method defines the world view of that method and thus specifies the limits of what can be modeled with a given method (Hirschheim et al. 1995). In other words, the method used for conceptual modeling defines the language and its grammatical rules that are used to articulate and communicate requirements by means of graphical models, and thus determines the outcomes of the modeling process, i.e., the quality of the model produced.

To date, many different modeling methods are available to practitioners, each with a different focus (e.g., data-oriented, object-oriented or process-oriented). Table 1 shows a few of the most popular approaches in use today.

Method	**Focus**	**Designers and proponents**
Data flow modeling	Describing the flow of data structures in partitioned systems	(Gane and Sarson 1979)
Entity-relationship modeling	Describing structures of data bases on a conceptual level	(Chen 1976)
Enterprise modeling	Describing and providing a graphical overview of the structure of organizations	(Checkland 1988)
Unified modeling	Describing systems in the form of encapsulated objects	(Booch 1999)
Process modeling	Describing business operations and the dynamics and behavior of information systems	(OMG 2009)
Object-process modeling	Describing both the structural and the dynamic aspects of a system via the building blocks object and process	(Dori 1996)

Table 1: Popular modeling methods

Despite the ongoing proliferation of modeling methods, we have seen a move towards a standardization of modeling approaches in recent years. And indeed, recent practitioner studies (Davies et al. 2006) indicate that two modeling approaches are in widespread use -- object-oriented modeling for software development (the primary application area of the UML standard), and process-oriented modeling for process documentation and improvement (the primary application area of the BPMN standard). The reported widespread use justifies our interest in UML and BPMN. We provide a short introduction to both approaches in the following sections.

Introduction to UML

The Unified Modeling Language (UML) has been in existence and use for approximately the last twelve years. It emerged in 1997 as a result of collaboration between its three proponents, Grady Booch, James Rumbaugh, and Ivar Jacobson, by combining their three independent modeling methods. UML has evolved through numerous modifications and revisions, with the current version being 2.2, according to the OMG (Object Management Group). The current version of UML contains 13 inter-related diagramming techniques for capturing and displaying information regarding models of systems. UML 2.x also adopted many more extension mechanisms than the original (Versions 1.x). The extension mechanisms add utility to the language but likely increase its complexity as a result, although this claim is conjecture at this point. According to several research efforts (Siau and Cao 2001; Siau et al. 2005; Dobing and Parsons 2006, Siau and Tian 2009), UML is quite complex. However, those research efforts also noted that UML is more often used informally and not as intended by its developers. Siau et al. (2005) introduced and coined the terms theoretical and practical complexity of modeling methods and argued that the use of modeling methods in the field might be very different from the use as advocated by the methods' developers. For example, UML's proponents insist that it be use-case driven, but, as Dobing and Parsons (2006) have noted, this may not be the actual case in practice. Regardless of how the language is used in the field, this research compares the complexity of UML (i.e., the language as a whole) with that of BPMN.

Introduction to BPMN

Business Process Modeling Notation (BPMN) is the result of a standardization effort that began in October 2001 as a working group within Business Process Management Initiative (BPMI), and was completed in its initial form in February 2006 under the auspices of OMG. BPMN was originally conceived as a graphical notation for the proposed Business Process Modeling Language (BPML) (Arkin 2002), but after work on BPML ceased in 2002, the focus of the working group shifted to provide a stand-alone notation with a mapping to the more popular Business Process Execution Language (BPEL) (Andrews et al. 2003). A first draft of BPMN was publicly released in August 2003, but work on the specification continued. In June 2005, the Business Process Management Initiative (BPMI) merged with the Business Enterprise Integration Domain Task Force of the OMG. A revised BPMN specification was sent to a finalization process within the OMG and published as an official OMG specification in February 2006. Updated specifications of the notation have been released in February 2008 (version 1.1 with extensions to the event symbols and a new signal event type, (OMG 2008)) and January 2009 (version 1.2 with editorial changes (OMG 2009)). BPMN is supported by more than 60 commercial and academic process modeling products and is finding rapid adoption in industry (Recker 2009). While BPMN follows an activity-centric notion of process modeling and shares certain symbols of UML Activity Diagrams (e.g., the rectangular Task symbol), it provides modeling

capabilities such as messaging, transaction management, and error handling, which go beyond the semantics of activity diagrams. Indeed, over recent years, process modeling with BPMN has emerged as a key instrument for the analysis and design of process-aware information systems (Dumas et al. 2005), service-oriented architectures (Rabhi et al. 2007), and web services (Ouyang et al. 2008) alike. Also, both BPMN and UML are part of the standard offerings of the Object Management Group, and anecdotal evidence suggests the intention of OMG to have these methods work in close collaboration and possibly the integration of the two methods (http://www.bpmn.org/Documents/FAQ-WG.htm).

While BPMN has found widespread adoption as a graphical notation, it was developed without a formal underlying meta-model, and did not provide a specification for a persistency format such as an XML schema or a file format. The BPMN 1.1 and 1.2 specifications provide a description of the BPMN modeling constructs using class-diagrams, but these diagrams do not express restrictions that govern the use of symbols (e.g., the rule that a message flow must not be used between elements that reside within a single pool, or the rule that a start event must not have any incoming sequence flows). The OMG attempted to remedy this by soliciting a separate meta-model that would simultaneously serve as a persistency format, but this effort was never finalized. Instead, the BPMN working group is currently developing a BPMN 2.0 specification that is said to contain both a formal meta-model and a persistency format, although no draft has been made publicly available to date.

The current BPMN version (1.2) contains 53 distinct graphical symbols that can be used to form process models at different levels of abstraction – ranging from high-level value-chain descriptions to detailed transactional processes that are designed to be embedded in applications. A previous analysis of BPMN modeling in practice (zur Muehlen and Recker 2008) has shown that a typical BPMN diagram contains less than 10 of these symbols, and that the frequency of symbol use follows a long-tail distribution, similar to the use of words in natural language. A possible conjecture that follows from this observation is that users deliberately reduce the complexity of the language by restricting its vocabulary (and consequently the rules governing the unused parts of the vocabulary).

Evaluation Techniques for Conceptual Modeling Methods

The increasing availability of different modeling methods has created an increased need for research to give guidance and insight to practitioners to assist them in the comparison, evaluation, and use of modeling methods (Moody 2005). Method evaluation is important so as to better understand the methods, and to assess the methods. Both researchers and practitioners want to have a better appreciation of the characteristics, strengths and weaknesses of methods in order to classify them, improve them, and enhance the information systems development process (Siau and Rossi 2009).

Siau and Rossi (2009) report on a comparison of techniques available to assist researchers and practitioners alike in the task of evaluating modeling methods. They broadly distinguish three types of techniques for evaluating modeling methods -- feature comparison, theoretical and conceptual investigations, and empirical evaluations. Feature comparison techniques typically involve checklists of ideal method features against which modeling methods can be compared (e.g., Yadav et al. 1988). While feature comparison has been predominant at some stage, subjectivity in design and conduct of the analysis has to be controlled.

Empirical evaluations (e.g., through surveys (Fedorowicza and Villeneuve 1999), laboratory experiments (Bodart et al. 2001), case studies (Fitzgerald 1997) and other methods require access to human subjects, case organizations or other forms of empirical data to establish knowledge about the methods by means of inductive logic, statistics or interpretive argumentation. Theoretical and conceptual investigations, by contrast, have developed means of evaluating modeling methods without the need of access to empirical data, and without problems relating to subjectivity of the analysis, by carrying out analytical studies in well-defined and narrow subject areas based on established terms of reference such as meta-models (Halpin 2002), ontology (Wand and Weber 1993), cognitive psychology (Siau and Tan 2005, Siau and Wang 2007) or linguistic (Siau and Tian 2009). Our interest in this paper lies in the conceptual investigation technique called metrics analysis.

Metrics analysis (Rossi and Brinkkemper 1996; Bajaj and Ram 1999; Marcos et al. 1999; Vanderfeesten et al. 2008) uses meta-model based metrics to evaluate the structural properties of a modeling method in order to make statements about the complexity of the modeling method. The premise in this approach is that the presence of more modeling elements increase cognitive load for the modeler (Chandler and Sweller 1991), thus making a method harder to learn and with more difficult rules to follow.

The Rossi and Brinkkemper (1996) complexity metrics use meta-models of modeling methods as a basis for analysis. In its simplest form, a meta-model is a conceptual model of a modeling method and captures information about the concepts, representation forms, and uses of a method (Siau and Rossi 2009). To enable the use of such meta-models for method analysis and comparison, Rossi and Brinkkemper (1996) developed a comprehensive set of complexity metrics that

quantitatively establish a method's complexity and ease of use based on the number of concepts and relationships defined in a method meta-model. These metrics are counts of objects, relationships, and properties, as well as the average connectivity of method elements and two overall metrics that position a method in a three-dimensional measurement cube. Rossi and Brinkkemper (1996) presented seventeen metrics clustered in three tiers -- individual, aggregate, and method-level metrics. While we could have used other metrics-based approaches (e.g., Bajaj and Ram 1999; Marcos et al. 1999; Vanderfeesten et al. 2008), the metrics by Rossi and Brinkkemper (1996) are an established and comprehensive set of metrics. Further, this set of metrics has in prior work been used to independently examine UML (Siau and Cao 2001) and BPMN (Indulska et al. 2009), as well as other modeling methods (e.g., Bajaj 2004; Zhang et al. 2007). Our intention in this paper is to compare the relative complexity of two modeling methods -- the process modeling method BPMN and the object-oriented modeling method UML.

RESEARCH APPROACH

To compare the complexity of BPMN and UML using the Rossi and Brinkkemper (1996) metrics, we refer to the details of our earlier individual complexity analyses as reported in (Siau and Cao 2001) for UML, and (Indulska et al. 2009) for BPMN. UML 2.x comprises thirteen diagram types including use class diagrams, class diagrams, sequence diagrams, timing diagrams, and others. This allows UML to specify both the structure and the behavior of information systems in conceptual models. To enable a meaningful comparison to the process-oriented modeling method BPMN, we only consider two UML diagram types that are intended to model behavioral aspects -- activity diagrams and state diagrams. The reason is that these two diagrams are closely related to BPMN. We complement this subset of UML by considering the overall complexity of the UML method as a whole.

For BPMN, we consider specification version 1.2 (OMG 2009) and adopt the BPMN complexity calculations presented in (Indulska et al. 2009) to facilitate the application of the metrics. The BPMN 1.2 meta-model differentiates abstract BPMN concepts (such as message recipient and flow objects) from their concrete, graphical instantiations (such as activity, message, and event). Using this differentiation, we consider two different sets of the BPMN method overall. First, we consider the complete BPMN meta-model as per the BPMN specification (OMG 2009). The complexity of the complete BPMN meta-model allows us to contrast the theoretical complexity (Siau et al. 2005) of BPMN with the theoretical complexity of the UML method as a whole. Second, we also consider the concrete BPMN set (as per the meta-model), i.e., the set of graphical instantiations only. This approach allows us to compare the complexity that could be encountered in graphical BPMN models with the complexity of the behaviorally oriented UML diagrams -- activity and state diagrams. In the following section, we discuss the results of this comparison.

FINDINGS

Table 2 below compares the complexity of the BPMN full and concrete meta-models with UML overall, and two of its diagramming techniques (i.e., Activity and State diagrams). We have done so for the following reason: BPMN is capable of modeling both activity-centric and state-centric processes. While the default BPMN use is comparable to UML activity diagrams in that dependencies between activities are modeled through the use of sequential flow and logical gateways, there are a number of BPMN modeling constructs that allow the modeling of state-transition-like diagrams. For instance, an activity can be embellished with boundary events that will terminate the execution of the activity once the specified event occurs. Thus, the activity behaves like a processing state, and the flow arrow linking from the boundary event symbol to the next activity represents the transition of a state-transition diagram. In a similar fashion, the event-based gateway behaves like a wait state, i.e., the BPMN process will remain in this state until a specified event occurs. Because of this potential dual use of BPMN, we are comparing the BPMN meta-model to both UML activity and state diagrams.

Method	Objects (Obj)	Relationships (Rel)	Properties (Prop)	Prop/Obj	Prop/Rel	Total Complexity
BPMN ^{FULL}	90	6	143	1.52	1.33	169.07
BPMN ^{CONCRETE}	57	6	74	1.19	1.33	93.60
UML ^{FULL}	57	53	72	1.26	1.36	106
UML Activity diagrams	8	5	6	.75	.2	11.18
UML State diagrams	10	4	11	1	0.5	15.39

Table 2: Complexity Metrics Evaluation Results

As can be seen from Table 2, the full BPMN model exhibits more complexity than the full UML meta-model. However, the concrete BPMN model appears to be similar to the UML full model. The important point to note is that for Rossi and

Brinkkemper's (1996) metrics, complexity correlates with the absolute number of elements in the language. In other words, the more elements that are included in the meta-model, the higher the measured levels of complexity.

The comparison of the complexity metrics for BPMN ^{FULL} versus BPMN ^{CONCRETE} further show that BPMN exhibits a high level of method complexity because of method constructs and constraints that are not graphically rendered (Indulska et al. 2009). This is indicated by the differences in the complexity measures of the full BPMN meta-model and its concrete subset (i.e., 169.07 vs. 93.60 in the full BPMN meta-model). This finding indicates that the underlying rules and constraints of the method are a significant source of complexity.

Although the complexity of full BPMN specification is higher than the full UML specification, the complexity of concrete BPMN specification is comparable to the full UML model. We also provide the complexity of activity and state diagrams in Table 2. Although BPMN is capable of modeling both activity-centric and state-centric processes, our findings suggest that BPMN is much more complex than UML activity and state diagrams. Indeed, inspection of Table 2 suggests that BPMN is significantly more complex than either Activity diagrams or state diagrams, in terms of object, relationships and properties, as well as the ratio between these components. From the analysis, an interesting question is whether process-oriented modeling is inherently much more complex (or difficult) to do than object-oriented modeling.

While our analysis indicates that the theoretical complexity of BPMN is higher than that of UML, we must note that the actual usage complexity of UML and BPMN may in fact be quite different and we do expect the complexity to be lower in practice. For the case of UML, for instance, Siau et al. (2005) found in their study that the theoretical complexity of UML is much higher than the practical complexity of UML. The reason is that not all the constructs are used all the times and not all the constructs are equally important. For BPMN, we expect a similar situation to manifest. Initial exploration of this expectation shows that similar situation exists for BPMN based on the complexity analysis of some commonly used BPMN subsets (Indulska et al. 2009). It may be that a large part of the theoretical complexity of BPMN is contained in elements that are rarely used in practice (see the published classification in zur Muehlen and Recker 2008). In practice, it is likely that much of the complexity is hidden from the users because the users do not have to pay attention to those rules when they do not use certain elements. This is the argument put forth by Siau et al. (2005) when they introduced the terms theoretical and practical complexity.

Our comparison makes clear that the complexity of UML is far from the complexity level its OMG cousin BPMN exhibits. In practice, however, BPMN has found widespread acceptance and wide-ranging tool support, similar to the case of UML – even though UML appears to be less complex in our complexity analysis. While our findings provide important information regarding the comparison of complexity of these two modeling methods, this situation would suggest that actual adoption and usage decisions are largely uninformed by the levels of complexity of a modeling method. Nevertheless, complexity is a trait of methods that can significantly inhibit developers' willingness to use a modeling method (e.g., Riemenschneider et al. 2002) and therefore warrants close attention in practice and research. Prior research on the trade-off between complexity and communication clarity (Gemino and Wand 2005) further suggests that in practice modeling method users would prefer efficient (i.e., non-complex) modeling methods over effective (i.e., highly expressive but complex) ones. However, one could argue that efficient modeling methods would be useful during analysis and an effective method would be important during design and development. Thus, the jury is still out on this issue and more research is warranted.

CONCLUSIONS

Our research indicates that while BPMN and UML are both complex methods, BPMN appears to be more complex than UML. However, when considering only the concrete graphical instantiations of the meta-models, the complexity of the two methods is relatively similar, at least from the metrical analysis we considered here.

Because both UML and BPMN are in relatively wide use, this indicates that developers have found ways to deal with the complexity. However, one thing to note is that not all constructs and rules are used in practice all the time (Siau et al. 2005, Erickson and Siau 2007). Empirical studies on the usage of BPMN (zur Muehlen and Recker 2008) and UML (Siau et al. 2005) confirm this statement. Thus, the practical complexity of BPMN and UML is expected to be lower than the theoretical complexity. This distinction is not only of relevance to practice but also an important note for future research on systems analysis and design. It will be key to arrive at an informed opinion not only about how a modeling method is theoretically possible to be used (the theoretical complexity), but also how it is actually being used (the practical complexity). Such information, furthermore, is an important aspect to consider in ongoing standards revision processes. Both UML and BPMN are currently undergoing revision and extension processes. Although such revision processes consider extensions and additions to published standards (e.g., Recker et al. 2007), our findings indicate that – in the interest of learnability, usability, and ultimately user acceptance – method developers should consider reducing the complexity of methods in their revisions, instead of striving to expand on the modeling expressiveness of methods.

Moving forward, we propose to continue our line of research to (a) explore the complexity of the component parts of UML and BPMN, and to (b) investigate the use of BPMN and UML in practice. This stream of research will allow us to develop a more comprehensive understanding of actual modeling practices, which, in turn, will increase our understanding of the factors that contribute to successful analysis and design of information systems. In a related stream of research, we also seek to provide normative guidance to researchers and practitioners on the question of how to deal with method complexity in actual usage. This could be achieved, for instance, by identifying the cores of modeling methods (e.g., UML and BPMN) as suggested by Erickson and Siau (2007). Such an analysis could then be complemented by assessing the ontological coverage (e.g., on basis of the work by Wand and Weber 1993) of the "common core" of each modeling method, and then examine the delta of ontological coverage of each added modeling element under consideration of its added complexity. Alternatively, an empirical investigation could study the trade-off between complexity gained versus ease of use in practice of different "common cores" of UML and BPMN.

REFERENCES

Ambler, S. W. The Object Primer: Agile Model-Driven Development with UML 2.0 (3rd ed.), Cambridge, England: Cambridge University Press, 2004.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S. Business Process Execution Language for Web Services. Version 1.1, 2003, available at http://xml.coverpages.org/BPELv11-May052003Final.pdf.

Arkin, A. Business Process Modeling Language, 2002, available at http://www.bpmi.org/.

Bajaj, A. "The Effect Of the Number of Concepts On the Readability of Schemas: An Empirical Study With Data Models," Requirements Engineering (9:4), 2004, pp. 261-270.

Bajaj, A., and Ram, S. "Evaluating Completeness of Conceptual Business Process Models: A Metric Based on Case Studies," Journal of Information Technology Cases and Applications (1:4), 1999.

Bodart, F., Patel, A., Sim, M., and Weber, R. "Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests," Information Systems Research (12:4), 2001, pp. 384-405.

Booch, G. "UML in Action," Communications of the ACM (42:10), 1999, pp. 26-28.

Chandler, P., and Sweller, J. "Cognitive Load Theory and the Format of Instruction," Cognition and Instruction (8:4), 1991, pp. 293-332.

Checkland, P. B. "Soft Systems Methodology: An Overview," Journal of Applied Systems Analysis (15:1), 1988, pp. 27-30.

Chen, P. P.-S. "The Entity Relationship Model - Toward a Unified View of Data," ACM Transactions on Database Systems (1:1), 1976, pp. 9-36.

Coad, P., and Yourdon, E. Object Oriented Analysis (2nd ed.), Englewood Cliffs, New Jersey: Prentice Hall, 1990.

Davies, I., Green, P., Rosemann, M., Indulska, M., and Gallo, S. "How do Practitioners Use Conceptual Modeling in Practice?," Data & Knowledge Engineering (58:3), 2006, pp. 358-380.

Dobing, B., and Parsons, J. "How UML is Used," Communications of the ACM (49:5), 2006, pp. 109-113.

Dori, D. "Unifying System Structure and Behavior Through Object-Process Analysis," Journal of Object-Oriented Programming (9:4), 1996, pp. 66-73.

Dumas, M., van der Aalst, W. M. P., and ter Hofstede, A. H. M. Process Aware Information Systems: Bridging People and Software Through Process Technology, Hoboken, New Jersey: John Wiley & Sons, 2005.

Erickson, J., Siau, K., "Theoretical and Practical Complexity of Modeling Methods", *Communications of the ACM*, (50:8), 2007, pp. 46-51.

Fedorowicza, J., and Villeneuve, A. O. "Surveying Object Technology Usage and Benefits: A Test of Conventional Wisdom," Information & Management (35:6), 1999, pp. 331-344.

Fitzgerald, B. "The Use of Systems Development. Methodologies in Practice: A Field Study," Information Systems Journal (7:3), 1997, pp. 201-212.

Gane, C., and Sarson, T. Structured Systems Analysis: Tools and Techniques, Englewood Cliffs, California: Prentice-Hall, 1979.

Gemino, A., and Wand, Y. "Complexity and Clarity in Conceptual Modeling: Comparison of Mandatory and Optional Properties," Data & Knowledge Engineering (55:3), 2005, pp. 301-326.

Green, P., Rosemann, M., and Indulska, M. "Ontological Evaluation of Enterprise Systems Interoperability Using ebXML," IEEE Transactions on Knowledge and Data Engineering (17:5), 2005, pp. 713-725.

Halpin, T. A. "Metaschemas for ER, ORM and UML Data Models: A Comparison," Journal of Database Management (13:2), 2002, pp. 20-30.

Hirschheim, R., Klein, H. K., and Lyytinen, K. Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations, Cambridge, Massachusetts: Cambridge University Press, 1995.

Iivari, J. "Why are CASE Tools not Used?," Communications of the ACM (39:10), 1996, pp. 94-103.

Indulska, M., zur Muehlen, M., and Recker, J. "Measuring Method Complexity: The Case of the Business Process Modeling Notation," in BPM Center Report 09-03: BPMcenter.org, 2009.
Jablonski, S. "On the Complementarity of Workflow Management and Business Process Modelling," ACM SIGOIS Bulletin (16:1), 1995, pp. 33-38.
Laudon, K. C., and Laudon, J. P. Management Information Systems: Managing the Digital Firm (9th ed.), Upple Saddle Revier, New Jersey: Prentice Hall, 2005.
Marcos, E., Cervera, J., and Fernandez, L. "Evaluation of Data Models: A Complexity Metric," in Siau, K. (Ed.) 4th Caise / IFIP 8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, Heidelberg, Germany, 1999.
Moody, D. L. "Theoretical and Practical Issues in Evaluating the Quality of Conceptual Models: Current State and Future Directions," Data & Knowledge Engineering (15:3), 2005, pp. 243-276.
Moody, D. L., and Shanks, G. "Improving the Quality of Data Models: Empirical Validation of a Quality Management Framework," Information Systems (28:6), 2003, pp. 619-650.
OMG Business Process Modeling Notation, V1.1, 2008, available at http://www.omg.org/spec/BPMN/1.1/.
OMG Business Process Modeling Notation, V1.2, 2009, available at http://www.omg.org/spec/BPMN/1.2/.
Ouyang, C., Dumas, M., ter Hofstede, A. H. M., and van der Aalst, W. M. P. "Pattern-based Translation of BPMN Process Models to BPEL Web Services," International Journal of Web Services Research (5:1), 2008, pp. 42-61.
Rabhi, F. A., Yu, H., Dabous, F. T., and Wu, S. Y. "A Service-oriented Architecture for Financial Business Processes: A Case Study in Trading Strategy Simulation," Information Systems and E-Business Management (5:2), 2007, pp. 185-200.
Recker, J. "Opportunities and Constraints: The Current Struggle with BPMN," Business Process Management Journal (15), 2009, p. In Press.
Recker, J., Indulska, M., and Green, P. "Extending Representational Analysis: BPMN User and Developer Perspectives," in Alonso, G., Dadam, P., and Rosemann, M. (Eds.) Business Process Management - BPM 2007, Brisbane, Australia: Springer, 2007, pp. 384-399.
Recker, J., Rosemann, M., Indulska, M., and Green, P. "Business Process Modeling: A Comparative Analysis," Journal of the Association for Information Systems (10:5), 2009.
Riemenschneider, C. K., Hardgrave, B. C., and Davis, F. D. "Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models," IEEE Transactions on Software Engineering (28:12), 2002, pp. 1135-1145.
Rossi, M., and Brinkkemper, S. "Complexity Metrics for Systems Development Methods and Techniques," Information Systems (21:2), 1996, pp. 209-227.
Siau, K., and Cao, Q. "Unified Modeling Language: A Complexity Analysis," Journal of Database Management (12:1), 2001, pp. 26-34.
Siau, K., Erickson, J., and Lee, L. Y. "Theoretical vs. Practical Complexity: The Case of UML," Journal of Database Management (16:3), 2005, pp. 40-57.
Siau, K., and Rossi, M. "Evaluation Techniques for Systems Analysis and Design Modelling Methods - A Review and Comparative Analysis," Information Systems Journal, 2009, p. in press.
Siau, K., and Tan, X. "Improving the Quality of Conceptual Modeling Using Cognitive Mapping Techniques," Data & Knowledge Engineering (55:3), 2005, pp. 343-365.
Siau, K., Tian, Y., "A Semiotics Analysis of UML Graphical Notations", *Requirements Engineering*, (14:1), 2009, pp. 15-26.
Siau, K., Wang, Y., "Cognitive Evaluation of Information Modeling Methods", *Information and Software Technology*, (49:5), 2007, pp. 455-474.
Vanderfeesten, I. T. P., Reijers, H. A., Mendling, J., van der Aalst, W. M. P., and Cardoso, J. "On a Quest for Good Process Models: The Cross-Connectivity Metric," in Bellahsene, Z., and Léonard, M. (Eds.) Advanced Information Systems Engineering - CAiSE 2008, Montpellier, France: Springer, 2008, pp. 480-494.
Wand, Y., and Weber, R. "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars," Journal of Information Systems (3:4), 1993, pp. 217-237.
Wand, Y., and Weber, R. "Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda," Information Systems Research (13:4), 2002, pp. 363-376.
Yadav, S. B., Bravoco, R. R., Chatfield, A. T., and Rajkumar, T. M. "Comparison of Analysis Techniques for Information Requirement Determination," Communications of the ACM (31:9), 1988, pp. 1090-1097.
Zhang, H., Kishore, R., Sharman, R., and Ramesh, R. "Agile Integration Modeling Language (AIML): A Conceptual Modeling Grammar for Agile Integrative Business Information Systems," Decision Support Systems (44:1), 2007, pp. 266-284.

zur Muehlen, M., and Recker, J. "How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation," in Léonard, M., and Bellahsène, Z. (Eds.) Advanced Information Systems Engineering - CAiSE 2008, Montpellier, France: Springer, 2008, pp. 465-479.