Towards a Usability Assessment of Process Modeling Languages

Kathrin Figl[†], Jan Mendling[‡], Mark Strembeck[†]
[†]Vienna University of Economics and Business (WU Vienna)
Augasse 2-6, A-1090 Wien, Austria
firstname.lastname@wu.ac.at
[‡]Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany
jan.mendling@wiwi.hu-berlin.de

Abstract: Visual process models provide support for analyzing and improving complex organizational processes. In this paper, we discuss differences of process modeling languages according to their ease of use and quality of user experience. These aspects are of importance for learning a modeling language, creating models, and understanding models. Modeling languages can be compared according to criteria as representational clarity, perceptual discriminability, perceptual immediacy, visual expressiveness, and graphic parsimony which are likely to influence cognitive effectiveness. In particular, we analyze the symbols sets of UML Activity Diagrams, YAWL, BPMN, and EPCs according to these criteria and discuss their strengths and weaknesses. Based on our findings, we propose a preliminary evaluation of the cognitive effectiveness of these modeling languages, which provides a basis for empirical studies.

1 Introduction

Business process models play an important role for the documentation of organizational processes and for the specification of information systems requirements. Although process modeling can be traced back at least to office automation research in the 1970s [Ell79], there is surprisingly little consensus on which elements should be covered by a process modeling language. Since the invention of EPCs in the early 1990s [KNS92, Sch00], there have been several other languages proposed including Petri nets [Mur89], UML Activity Diagrams [OMG09b], YAWL [AH05], and BPMN [OMG09a]. Yet, the preference for a particular language or the intervention of yet another language is often (not always) based on claims about usability (see [FLM$^+$09]). However, usability considerations provide a good angle for checking the superiority of a language from an empirical perspective.

An important reference discipline to discuss usability issues of process modeling languages is cognitive research. This area deals with insights on how humans process information, create knowledge, and solve problems. A central property of the human brain is that it works with specialized areas. Sensory information from eyes and ears is transmit-

ted to the *sensory memory*, which holds an after-image of that information for a moment. This after-image is analyzed and key concepts are held in *short-term memory* for some minutes. *Visual perception* enables understanding of sensory information. Those aspects that can be linked to other prior knowledge can become part of *long-term memory*. All these three different memory parts have different characteristics, which are the foundation of various theories of comprehension, learning, and problem solving. Several of these theories have been applied to information systems modeling, among others cognitive load theory [Swe88], cognitive fit theory [Ves91], the cognitive dimensions framework for notational systems [GP96], and the theory of multimedia learning [May01]. All of them identify perspectives that can also be considered for discussing advantages and drawbacks of process modeling languages.

There are many orthogonal characteristics of process modeling languages that can be subject to a cognitive investigation. In this paper we focus on a recent framework proposed by Moody and Van Hillegersberg [MH08] who discuss desirable properties of notation elements from a perceptual perspective. Such a perspective is particularly appealing for process modeling languages since many of the prominent languages share a rather large common subset. Different workflow pattern analyses have shown (see, e.g., [Men08]) that UML Activity Diagrams, YAWL, BPMN, and EPCs share actions and different routing elements including different types of splits and joins. Our contribution is a discussion of how the principles of Moody and Van Hillegersberg can be used to identify strengths and weaknesses of process modeling languages. Furthermore, we synthesize them in terms of propositions, which speculate on the cognitive effects of these strength and weaknesses. In this way, we pave the way towards an empirical investigation of their arguments in an experimental setting.

The remainder of this paper is structured as follows. In Section 2, we summarize the major arguments on usability of modeling languages. Section 3 discusses cognitive theories which are relevant for understanding process modeling languages. Section 4 then turns to the work by Moody and Van Hillegersberg, which we utilize in Section 5 to discuss strengths and weaknesses of process modeling languages. Finally, Section 6 concludes the paper and gives an outlook on future research.

2 Theoretical Background

Usability in general can be described as the measure of the ease of use and quality of the user experience when interacting with a device the user can operate in some way or another [Nie09]. Similarly, according to the ISONORM 9241/11, usability is defined by the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (see [Int98]). In the context of modeling, *effectiveness* means how well the modeling technique assists in accomplishing modeling goals and *efficiency* refers to the resources needed to use a particular modeling technique (see [GW04]). Since the benefit of using a specific language for communication between analysts, developers and users largely depends on the cost of production, learning curve and speed of decay [AEHE07], usability aspects of modeling

languages are of practical relevance. Moody [Moo05] notes that in the area of researching the quality of modeling languages there is lack of recognition of the relevant ISO standards for quality management and suggests to build on these foundations. ISONORM 9241 is a multi-part standard entitled 'Ergonomics of Human System Interaction' and in the sub part 9241/110 general usability heuristics for the design, the description and evaluation of interactive systems are proposed [Int06]. Although these principles were designed for the interaction between a user and a software system, at least the criteria: *suitability for the task*, *self-descriptiveness*, *conformity with user expectations*, *suitability for individualization* and *suitability for learning* also fit to the evaluation of modeling languages (*controllability* and *error tolerance* fit less). Nielsen [Nie09] refers to similar quality characteristics for usability: *learnability*, *efficiency* (time needed to perform a task), *memorability* (performance after a period without using it), *amount of errors made by users* and *user satisfaction*. When considering the usability of a modeling language, *ease-of-use* refers to the effort of learning the language, using the language (active modeling) and understanding models defined in the respective language [RR07, p. 2022]. Although there are standards for measuring quality and usability of software, there is no equivalent for models and modeling languages [Moo05]. Therefore, the next section will detail a number of factors that have to be take into account for an usability evaluation of modeling languages.

3 Cognitive Aspects of Modeling Languages

Basically, two main aspects of modeling can be distinguished (see [GW04]):

- **Creating models:** Process aspects of creating models include the effort (e.g. time) required to construct them, subjective ease-of-use, and ease-of-learning. The result of the modeling process is a model, which can be assessed concerning different quality aspects (such as correctness, accuracy, detail, completeness, quality, type of errors).
- **Understanding/Interpreting models:** In the process of model understanding, much cognitive activity is involved. The outcome of understanding is cognitive per se, i.e. it is created in the viewer's cognition and not directly observable. It can only be measured indirectly, e.g. via 'problem-solving' tasks in which newly developed mental models have to be used or via comprehension tests (see, e.g., [GW04]). Additionally, recall tests or correctly identified problems can be used to capture 'understanding'. Models from different modeling languages are likely to differ according to the effort required to interpret them and develop understanding, as well as in the perceived difficulty of obtaining information through their visual representation.

Both of these aspects show the complex interplay between human cognitive models and visual models. Therefore, the ability of a language to support appropriate translations between cognitive and visual models is probably the most essential criterion for determining it's usability. Norman's theory of action [ND86] can be used to explain differences between cognitive models of a model creator/viewer and a visually expressed model [GW04,

p. 256].

- A ‘gulf of execution’ in modeling is the gap between a user’s goals and the means that are applied to execute that goal. Discrepancies may, for instance, result from a modeling language that is not expressive enough and lead to extra effort, i.e. requiring additional thinking and distract from achieving the original modeling goal.
- A ‘gulf of evaluation’ refers to the gap between a stimulus (a model) and the resulting comprehension. Comprehension problems causing deviations between cognitive models and the actual visual representation/the intended meaning can for example be provoked by ambiguity of a modeling language.

Cognitive theories are central to the usability evaluation of different modeling languages. Mental processes as visual perception, information processing, reasoning and problem solving, attention, as well as short and long term memory are affected in learning how to use specific modeling languages, creating models, and understanding models. Aside from the fact that e.g. learning a modeling language may demand different mental processes than understanding a model, we hypothesize that there are specific aspects inherent a language that are likely to have an impact on all of these cognitive processes.

Up to now a variety of underlying cognitive theories were adopted to the context of modeling - often to explore potential benefits of the visual expression - as the dual coding theory [Pai91], the theory of working memory [BH74], the cognitive load theory [Swe88], the theory of multimedia learning [May01] and the external cognition theory [SR96].

Since process models usually represent complex relationships and humans have limited information processing capabilities (see [Ves91, p. 220]), a main goal in the design of process modeling languages is to reduce cognitive load for users to enable more effective problem solving. Low cognitive effort is positively related to a variety of quality aspects of models (see [ST05, p. 352]). Maes et al. [MP07] even define the perceived ease of understanding of a model as “the degree to which a person believes that using . . . would be free of mental effort”. Cognitive load is determined by the amount of elements needed to be paid attention to at a point of time [Kir02]. There is a natural limit of the capacity of short-term memory of humans of approximately 7 +/- 2 elements [Mil56]. Although the amount of elements is limited, their size and complexity is not. Chunking expands the capacity of short-term memory, because information units belonging together are chunked into one unit [GLC$^+$01]. Short-term memory, refined by Baddely and Hitch [BH74] as working memory, includes besides a central executive component a phonological loop and a visuo-spatial sketchpad. Performance on simultaneous tasks requiring separate perceptual domains (as visual and verbal) is much higher than for example working with two different texts. Similarly, the dual coding theory postulates that visual information (e.g. pictures) and verbal information (e.g. texts) are stored and processed differently via separate mental channels that do not compete with each other (see [Pai91]). In particular, research found evidence that learning material combining text and visual images support learning better than text only. The cognitive multimedia learning theory [May01] proposes different principles how to combine text and pictures. According to the *contiguity principle*, learning is better, when text and pictures are presented spatially near each other. In

consequence, text and icons belonging together should be placed near each other in visual models. However, the combination of text and visual information can also lead to cognitive overload (*split-attention principle*).

The cognitive load theory [Swe88] details how the working memory load influences learning and knowledge acquisition. In experiments, Sweller [Swe88] found out that heavy cognitive load during problem solving exercises impairs learning, especially schema construction. The theory differs between three types of cognitive load: *intrinsic*, *extraneous* and *germane cognitive load*. In contrast to *intrinsic cognitive load* (determined by the complexity of information, i.e. the amount of elements, and their relations and interactions), the *extraneous cognitive* load is influenced by the way the information is represented (see [Kir02, p.4]). While the cognitive load devoted to learning and understanding (*germane cognitive load*) should be promoted, *extraneous cognitive load* should be held low. This can, for example, be achieved by reducing additional, irrelevant information. By mapping the cognitive load theory to the context of modeling, it becomes clearer, why modeling languages might vary in their cognitive effectiveness. If the same information is modeled in different modeling languages, the resulting models should, to a large extend, have a similar *intrinsic cognitive load*, but they differ in their *extraneous cognitive load*. The amount of *extraneous cognitive load* caused by the modeling languages leads to differences in learning and understanding (see, e.g., [CS96]).

Due to the complex control flows, the creation and understanding of process models is likely to demand high cognitive reasoning and effort for logical thinking for human users. Additionally, users may, for example, want to deduce correct assumptions on how business processes are carried out in an organization based on a process model, or determine what will happen if a process cannot be carried out. Visual models not only demand, but also support users in their reasoning processes because "cues to the next logical step in the problem may be present at an adjacent location" [LS87, p.65]. Most research on deductive reasoning focuses on experiments with textual material and little is known about reasoning involved in visual modeling. Research, has shown, for instance, that there exist systematic fallacies (so called 'illusory inferences') when individuals internally construct mental models on premises including modeling-level connectives (like conjunctions, inclusive, or exclusive disjunctions) and draw conclusions from these mental models (see, e.g., [KJL09, NJL06]). In correspondence to mental models this may also hold true for externalized visual process models. In addition, the vast body of literature on error analysis of process models seems to give a hint on the occurrence of systematic reasoning fallacies concerning connectors (see, e.g., [MNvdA07, MvdA07, Men07]). Since correct interpretation of connectors in process models is inevitable for overall understanding, we hypothesize that the visualisation of connectors may especially affect whether different process model notations support understandability.

Besides creating and understanding models, learnability is a main factor for the usability of a modeling language. Learnability of the modeling notation largely depends on the depth of knowledge someone wants to acquire (e.g. learning the full set of language constructs, vs. learning a subset that is sufficient for conducting a specific modeling task). Green and Petre [GP96] point out that 'consistency' of visual languages is relevant. This means that users who have learned a certain part of the language can infer the rest. In more complex

languages, as for example BPMN (39 language constructs), it will take longer to acquire expertise, as in more simple languages, such as EPC (9 language constructs) [RD07, p. 358]. Similar to notations in physics or geometry some knowledge on the conventions of the notation will be inevitable for understanding the intended meaning of a process model. Other general aspects regarding learnability include the prior knowledge and experience or the instruction, which may have more impact on learning than the language per se. Recker and Dreiling [RD07, p. 351] showed that transfer concerning the ability to understand and read a model from a specific process modeling notation to another is surprisingly easy. Students who had solely be trained to EPC models scored equally well in comprehension tests of BPMN and EPC models.

4 Evaluating Cognitive Quality of Modeling Languages

The form of visual information representation can have a significant impact on the efficiency of information search, explicitness of information, and problem solving (see [LS87]). Compared to the semantic development, little research has been undertaken in order to improve the visual syntax of modeling languages (see, e.g., [MH08]). One of the key goals for the visual design of a model is that viewers draw attention to those components crucial for understanding and cognitive inferencing (see [SR96, p.207]). According to Moody and Hillersberg [MH08] there are 5 principles for cognitive effective visual design of modeling notations: semiotic clarity, perceptual discriminability, perceptual immediacy, visual expressiveness and graphic parsimony.

- **Semiotic/Representational Clarity**: This principle points out the importance of a good fit between the graphical symbols used in a modeling notation and the semantic concepts they refer to. Anomalies like symbol redundancy (more than one symbol represents the same concept), overload (one symbol represents more than one concept), symbol excess and deficit (there are graphical symbols without a correspondence to a semantic construct or vice versa) should be avoided, since they lead to ambiguity and additional unnecessary cognitive load for the user (see [MH08, p. 20]). For cognitive efficiency, graphical constraining will be a goal[1]. Consequently, notations will also highlight specific aspects of information at the expense of other aspects (see [GP96]). For evaluating the representational potential of a process modeling language the Bunge-Wand-Weber model [YR95] can be used. In a recent comparative analysis (see [RRIG09]), the modeling techniques EPC and BPMN were compared concerning their ontological completeness and clarity. The results of this comparison revealed interesting differences that are likely to have an effect on the cognitive efficiency of these languages.
- **Perceptual Discriminability**: Perceptual discriminability of symbols determines how easy it is for a user to distinguish between different symbols and visually rec-

[1] Graphical constraining is defined by Scaife and Rogers [SR96, p.189] as "the way graphical elements in a graphical representation are able to constrain the kinds of inferences that can be made about the underlying represented world"

ognize differences between them. It is highly influenced by the amount of visual variables in which symbols differ (also referred to as visual distance). If visual symbols are highly unique on a visual variable, they are likely to 'pop out' and are easy to locate in a model [MH08, p. 21]. Low perceptual discriminability can lead to misunderstandings. Research showed that for instance rectangles and diamonds in ER diagrams are easily confused (see [NC99]). On the other hand, if different symbols in a notation have similar attributes as color or shape, they are likely to be recognized as belonging together. According to the 'Gestalt law of similarity', elements will be grouped together if they have similar attributes (see [Wer38]). In consequence, symbols in a modeling language should differ sufficiently by visual variables to be perceptual discriminable. However, sometimes it is intended that symbols share visual variables if they should be recognised as related to each other.

- **Perceptual Immediacy**: Perceptual immediacy supports the user's understanding of the meaning of graphical symbols and representations, and describes whether symbols and their corresponding concepts are easily associated. Icons, for example, are easily associated with their referent real-world concepts. Iconic representations for classes of activities could improve the understandability of process models as suggested by [MR08], but are not yet commonly used. Additionally, spatial relationships of symbols can help to induce specific desired interpretations by a user (e.g. left-to-right implies sequence). Even a small layout change in the same graph may transport a different meaning (e.g. centrality vs. hierarchy) (see [AEHE07]). Both principles, iconic and spatial representation, seem to be related to the concept of cognitive metaphors [Lak92], which refers to the mapping of a conceptual domain in terms of another target domain. Modeling languages are likely to differ according to the intuitiveness of the visual metaphors they use for expressing real-world concepts. Theories of long term memory as the spreading activation theory [And83] assume that internal mental representation consist of associative networks in the form of a graph. According to Rockwell and Bajaj [BR05], modeling effectiveness and efficiency will be higher if the symbols used in a modeling notation are more similar to "the concept of node-relationship arc depiction of information". Therefore, it can be hypothesised that process modeling notations with nodes and edges are likely to be intuitively understandable (because of their compatibility with internal mental representations).

- **Visual Expressiveness**: Modeling notations which fully exploit the range of visual variables (spatial dimensions like horizontal and vertical, as well as shape, size, colour, brightness, orientation, and texture) have higher visual expressiveness. In comparison to a textual representation (words), which are encoded verbally in their reading direction, visual symbols are internally encoded in their spatial arrangement (see [San77]). Therefore, it is of importance to assess spatial dimensions of modeling notations. Using swimlanes in activity diagrams, for example, includes both planar variables for depicting information on who is working on an activity (see [MH08, p. 29]).

- **Graphic Parsimony**: Model complexity is determined by "the use of embedded symbols and distinct graphic symbols" [NC99, p. 141]. High graphic complexity of

a modeling notation may impair understanding and therefore should be avoided. On the other hand there will be a tradeoff between complexity and expressiveness.

5 Language Differences

5.1 Basic Elements of Process Modeling Languages

In general, process modeling languages include a number of basic, generic, and consensual elements that allow to define process flows. Below we briefly introduce these basic elements, before the subsequent sections discuss how different process modeling languages define these generic elements.

Start A start node defines the initial node of a process. A start node may have one or more outgoing edges.

End An end node defines the final node of a process or sub-process. An end node does only have incoming edges.

Tasks Different variants of task nodes exist. In general, a task node models a clearly defined action or process step. Tasks can be conducted by human users or software agents.

Decision A decision node models choices in a process flow. Decision nodes thereby define alternative routing options in a certain process.

Merge A merge node consolidates different (optional) paths that result from a choice made by a decision node.

Split A split node models parallel branches in a process. Thus, sub-processes started by the same split node are performed simultaneously.

Join A join node synchronizes parallel sub-processes. Typically, a join node synchronizes sub-processes resulting from a split node.

Event An event defines the occurrence of a certain type of 'incident'. In process models events are typically used to trigger tasks. In addition, a task may also produce events which may trigger other tasks.

In Figures 1 and 2, we see five variants of a simplified credit application process modeled via an UML activity model, BPMN, YAWL, EPC, and as classical Petri net, respectively.

5.2 UML Activity Models

Figure 3 depicts the basic elements of UML2 activity models (see [OMG09b]). The main element of an activity model is an Activity. The Activity represents a process that consists of Actions and different types of control nodes. Actions thereby define the tasks

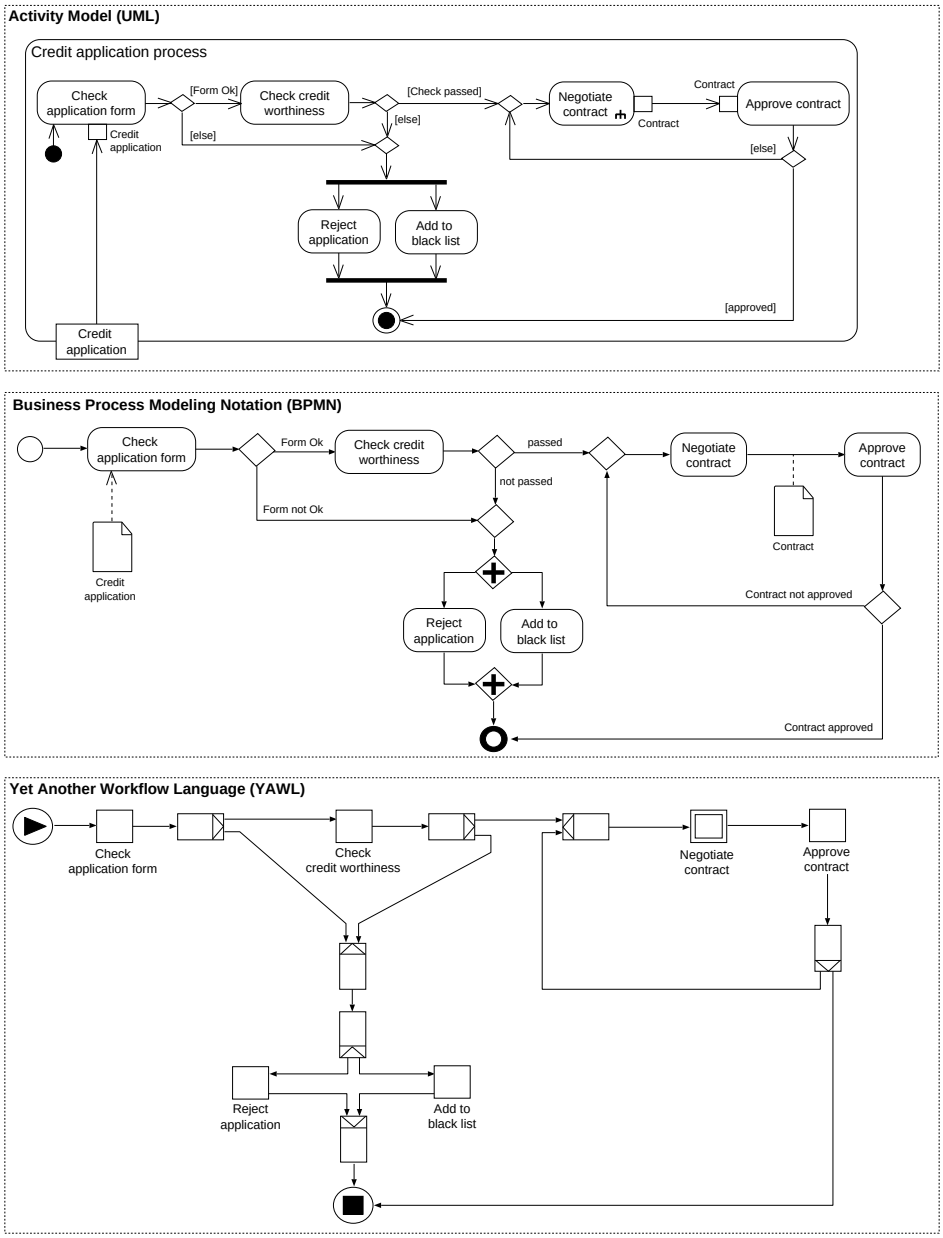

Figure 1: An example process modeled via UML, BPMN, and YAWL

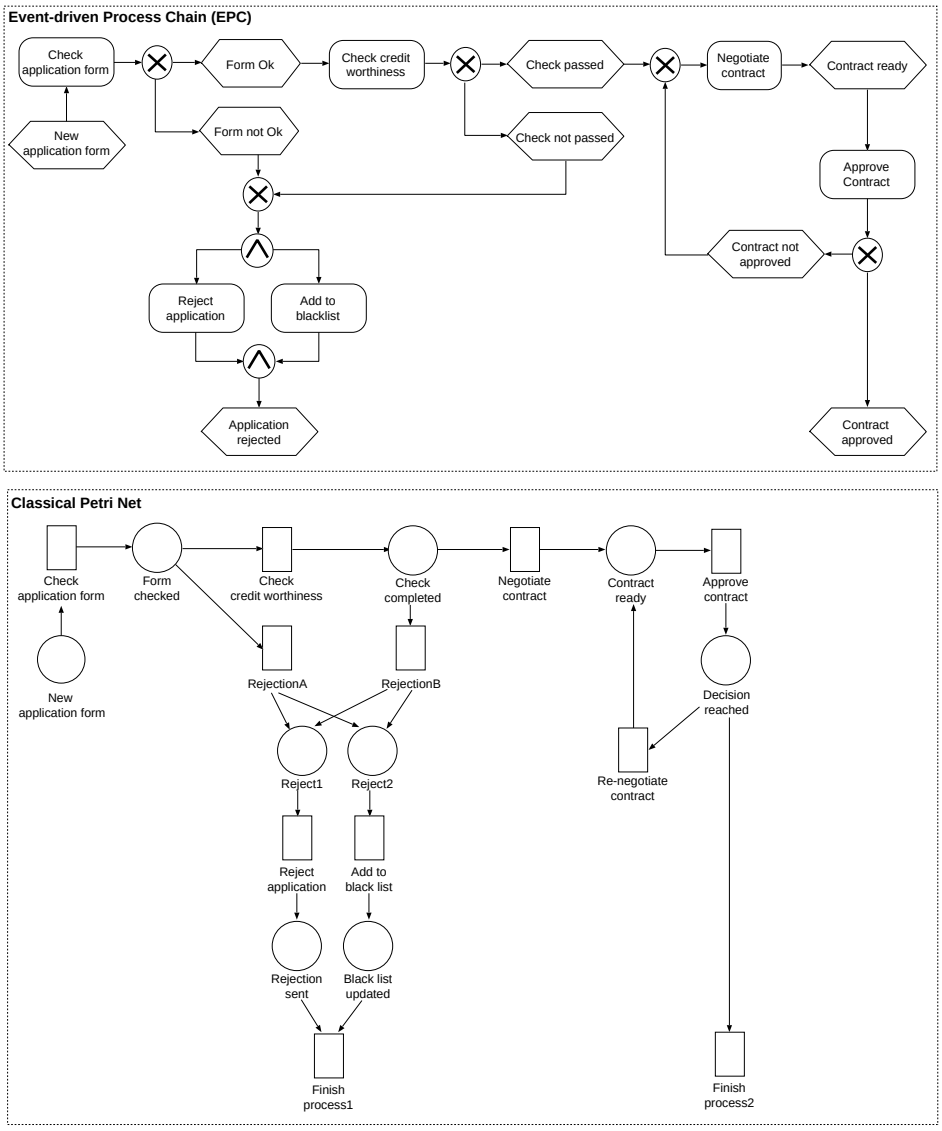

Figure 2: An example process modeled as EPC and classical Petri net

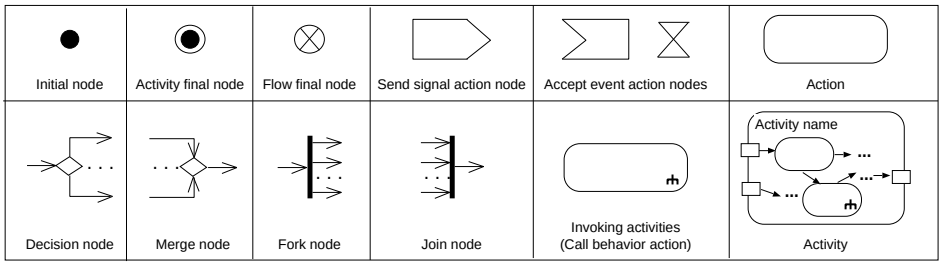

Figure 3: Basic elements of UML activity models

(steps) that are performed when executing the corresponding Activity. To model the (synchronous or asynchronous) invocation of other processes from an Activity, one may use an action symbol (round-cornered rectangle) including a small rake symbol (see Figure 3). In essence, this modeling element defines that the corresponding step in a process is itself a process that is modeled via an own Activity model.

Activity models have a token semantics, similar (but not equal) to Petri nets. Each Activity may have one or more Initial nodes and one or more Final nodes. In case the process flow reaches an Activity Final node, it immediately stops the execution of all actions and terminates the respective Activity. In contrast, a Flow Final node only destroys the tokens flowing into it but does not terminate the Activity. A Decision node is represented by a diamond-shaped symbol and has one incoming and multiple outgoing edges. A Merge node is represented by a diamond-shaped symbol and has multiple incoming and one outgoing edge. A Fork node is represented via a thick line and has one incoming and multiple outgoing edges. A Join node is represented via a thick line and has multiple incoming and one outgoing edge. A Send Signal node is represented by a convex pentagon. Two different types of Accept Event Action nodes exist. An ordinary Accept Event Action node is represented via a concave pentagon and waits for the occurrence of a particular type of event. The corresponding event type may result from a signal produced by a Send Signal node. An Accept Time Event Action node is represented by an hour-glass shaped symbol.

The UML notation partially violates some of the criteria identified by Moody. There is some gap of representational clarity with different end symbols (activity final and flow final). On the other hand, the different end symbols have clearly defined and distinct semantics. Furthermore, start and end nodes look quite similar. A strong point of UML is the clear discrimination between decisions and concurrency. Both routing elements are significantly different. A certain degree of immediacy is provided by Send Signal and Accept Event nodes. The rest of the notation remains rather abstract. Visual expressiveness is used to distinguish concurrent from alternative branching. Altogether, the notation set is quite parsimonial although it is quite expressive.

5.3 Business Process Modeling Notation (BPMN)

Figure 4 depicts the core process model elements of the BPMN notation [OMG09a]. Note that there are further symbols, which are more specific, e.g. for throwing exception events. BPMN uses different types of events for defining the start and end of a process. The symbols shown in the figure can be extended with additional symbols to describe, for instance, a message start event or a timer. BPMN also distinguishes different types of end events. The terminate event explicitly kills all parts of the process that still might be active. Sending and receiving intermediate events are defined by filled and unfilled symbols within a circle, respectively. The different steps in a process are captured as tasks. Different types of tasks exist, e.g. embedded and collapsed sub-processes. Finally, there are gateways for specifying routing constraints. Splits and joins both use a diamond shape. XOR gateways can be drawn without a symbol or with an X inside. AND gateways contain a plus sign.

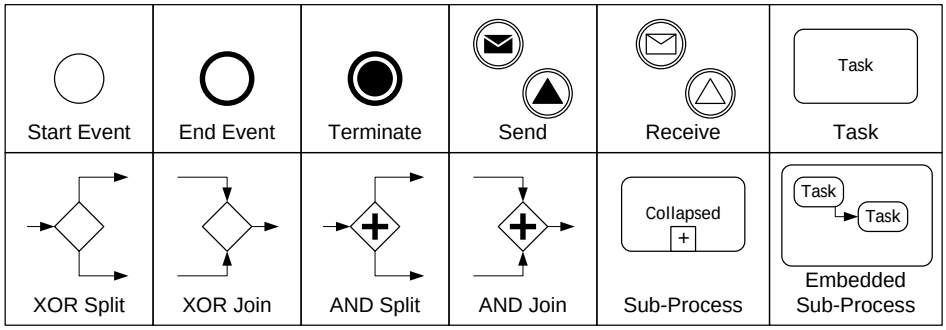

Figure 4: Basic elements of BPMN diagrams

The BPMN notation exhibits different weaknesses. Representational clarity is partially violated. XOR gateways can be described with different symbols (blank diamond and X diamond). This maybe confusing to model readers. Furthermore, there is little perceptual discrimination between different pairs of elements. Start, end, and intermediate events are only distinguished by their border (thin, bold, double-lined). Also XOR gateways as X diamonds are difficult to distinguish from AND gateways with a plus. This may lead to confusion, too. Intermediacy is rather low with the blank set of symbols. On the other hand, many event subtype symbols (e.g. envelop for message event) are very intuitive. Visual expressiveness is somewhat limited for the standard set of symbols: tasks and gateways are both quadrangles and gateways are circles. However, a rich set of shapes is used as symbols for event subtypes along with varying brightness for throwing (filled black) and catching of events (not filled). Texture and color is not used in the standard symbol set, but is often introduced by modeling tools for highlighting tasks. Finally, there is hardly any graphic parsimony. BPMN uses an extensive set of symbols and subtype symbols. This might make it difficult to learn the language.

5.4 Yet Another Workflow Language (YAWL)

The core elements of YAWL are shown in Figure 5. YAWL uses a unique start and end condition to mark the beginning and termination of a process. Conditions are used to separate tasks (similar to places in Petri nets), yet they do not need to be explicitly drawn. Tasks are depicted as rectangles. There are variants of tasks, e.g. multiple instance tasks and sub-process tasks. Routing behavior is defined by thin blocks on the left-hand and right-hand side of tasks. Both XOR and AND splits and joins use triangle symbols.

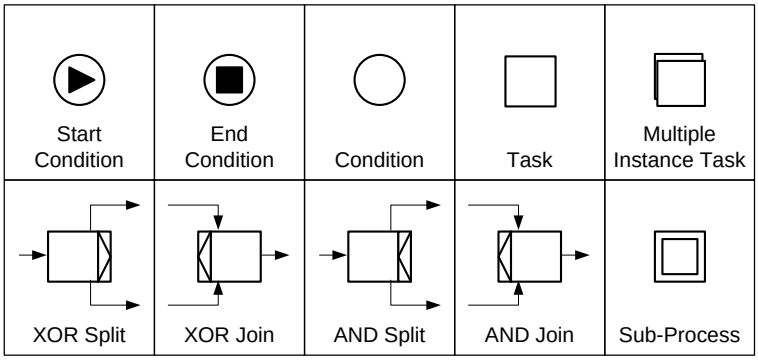

Figure 5: Basic elements of YAWL diagrams

The criteria by Moody point to potential problems of the YAWL notation. Representational clarity is violated, because a decision and a simple merge can be described either by conditions with multiple incoming and outgoing arcs, or with XOR joins and splits. There is also no direct support for events in YAWL. Thus, catching and throwing of events have to be defined using tasks. There is also a serious problem with discrimination of AND and XOR splits and joins. They share the same symbol. The semantics have to be identified also based on the position on the split/join block along with the number of incoming and outgoing arcs to the respective task. YAWL is also a quite abstract notation. However, start and end conditions provide a good intuition of their semantics based on an audio player metaphor. There is little visual expressiveness due to little variation in shape and size. Filled shapes are only used for starts and ends. On the other hand, the notation is quite parsimonial, at least when compared to BPMN.

5.5 Event-driven Process Chains (EPCs)

The EPC notation contains three types of symbols: circles for connectors, rounded rectangles of functions, and hexagons for events. There is no distinction symbol-wise between different types of events. Different connectors can be distinguished by their symbol, $\times$ for XOR, $\wedge$ for AND, and $\vee$ for OR. Beyond that, there are two ways of defining sub-processes, either by a sub-process symbol on the right bottom corner of the function symbol or as a so-called process interface to start a subsequent process.

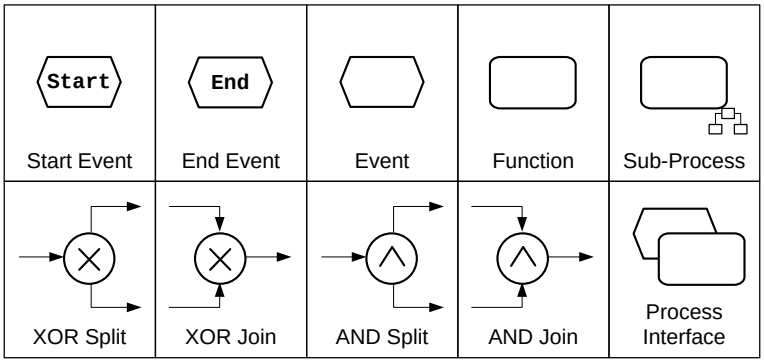

Figure 6: Basic elements of EPCs

EPCs also have problems with different usability criteria. The event symbol is heavily overloaded, which causes problems in terms of clarity. Start, end, intermediate events, and all kinds of events types map to the same symbol. There is also a problem to easily discriminate AND and OR connectors. They share the same symbol, only mirrored vertically. Altogether, the symbol set of EPCs is very abstract, such that there is little immediacy and intuition of the symbols. Visual expressiveness is also limited. Some tools depict events and functions in red and green color, which helps to distinguish them. The notation is also parsimonial by using only a small symbol set.

5.6 Petri Nets

Petri nets have the smallest symbol set of the languages discussed here. Transitions capture tasks, as well as synchronization (multiple in-arcs) and introduction of concurrency (multiple out-arcs). Places define a partial state, that can also be used to define decision points and simple merges.

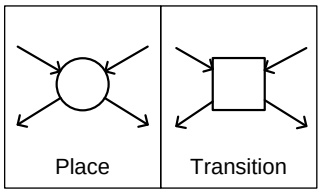

Figure 7: Basic elements of Petri nets

Due to the small set of symbols, Petri nets have problems with clarity. Their elements capture quite different behavior depending on the number of incoming and outgoing arcs. The two symbols, rectangles for transitions and circles for places, can be easily discriminated, though. Yet, they remain abstract such that there is no intuition or immediacy. Visual expressiveness only builds on the parameter shape, which is OK for only two sym-

bols. Altogether, Petri nets are a language of extreme graphic parsimony using only two different symbols.

5.7 Evaluation and Discussion

Based on the number and types of issues identified above we can hypothesize about understanding and learning performance depending on the process modeling language.

Representational Clarity A lack of clarity is likely to result in two problems when creating a model based on redundant symbols. First, modelers might need more time to decide which among several alternative elements to use for representing a certain real-world fact. Second, models created by different modelers may vary significantly in terms of the symbols they use for the same facts. These problems are likely to show up with BPMN and YAWL. On the other hand, symbol overload may result in corresponding ambiguity for somebody who is reading the model. This issue seems to be more relevant to EPCs and Petri nets. UML activity diagrams seem to have the highest degree of clarity from the languages being discussed.

Perceptual Discriminability Good discrimination should support fast visual perception and good recall in short term memory. All languages have some weaknesses in discriminating start and end nodes except YAWL. On the other hand, YAWL is very weak in discriminating routing elements. This is also partially the case for EPCs (OR versus AND) and BPMN (X versus +). UML very clearly distinguishes concurrency and alternative branching with significantly different symbols. Finally, EPCs have some weaknesses in discriminating functions and events.

Perceptual Immediacy Many language elements are very abstract such that no intuition is provided. Some counter examples are start and end conditions in YAWL and certain event types in BPMN. These elements should be easy to identify and to remember.

Visual Expressiveness The visual expressiveness of process modeling languages is limited. Most elements only vary in shape and size. Counter examples are again events in BPMN (throwing versus catching), YAWL start and end, as well as UML concurrency and alternative routing.

Graphic Parsimony BPMN is by far the richest language in terms of its symbol set. Therefore, it should also be the most time consuming to learn. YAWL and UML also provide a rather large set of symbols. EPCs and Petri nets are lean in terms of symbols, such that they should be easy to learn notation-wise. On the other hand, they are the most abstract notations such that they should be the most difficult ones to be applied correctly.

The different observations on weaknesses of process modeling languages offer the chance for a comparative evaluation. A particular aspect that is strongly supported by one language and weakly by another should directly materialize in a difference in understanding

or learning performance. A respective empirical experiment is missing so far, and on our agenda of future research.

6 Conclusion

Despite increasing consciousness about the need to consider the (non-technical) user's point of view in the area of information systems engineering, little research has been undertaken in order to improve and understand the usability of modeling languages. Up to now, few studies have investigated differences of process modeling languages concerning human factors such as ease-of-use, understandability, or learnability. On that account we build on cognitive theories to identify different perspectives for discussing usability of modeling languages. The main contribution of this paper is an analysis of the symbol sets of the five most prominent process modeling languages according to characteristics which are likely to have an impact on cognitive effectiveness and usability. The evaluation results show remarkable differences of languages according to representational clarity, perceptual discriminability, and graphic parsimony. Consequently, to meaningfully examine and better support the choice and development of process modeling languages, future research should investigate whether these differences of the visual notations in fact influence understandability or learnability. Further empirical studies including human errors analysis as well as problem-solving, comprehension, and recall tasks have the potential to provide new insights concerning the usability of different process modeling languages and to test our preliminary hypotheses.

References

[AEHE07] J. Aranda, N. Ernst, J. Horkoff and S. M Easterbrook. A Framework for Empirical Evaluation of Model Comprehensibility. In *29th International Conference on Software Engineering (ICSE'07). International Workshop on Modeling in Software Engineering (MiSE-07)*, Minneapolis, USA, 2007.

[AH05] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4), June 2005.

[And83] John R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 1983.

[BH74] A.D. Baddeley and G. Hitch. *The psychology of learning and motivation: Advances in research and theory*, Jgg. 8, Kapitel Working memory., pp. 47–89. Academic Press, New York, 1974.

[BR05] Akhilesh Bajaj and Stephen Rockwell. *Advanced Topics in Database Research*, Kapitel COGEVAL: A Propositional Framework Based on Cognitive Theories To Evaluate Conceptual Models, pp. 255–282. Idea Group Publishing, 2005.

[CS96] Paul Chandler and John Sweller. Cognitive Load While Learning to Use a Computer Program. *Applied Cognitive Psychology*, 10(2):151–170, 1996. 10.1002/(SICI)1099-0720(199604)10:2¡151::AID-ACP380¿3.0.CO;2-U.

[Ell79] C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pp. 225–240, Boulder, Colorado, 1979. ACM Press.

[FLM+09] D. Fahland, D. Lübke, J. Mendling, H. Reijers, B. Weber, M. Weidlich and S. Zugal. Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. In Terry Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer and Roland Ukor, Hrsg., *Enterprise, Business-Process and Information Systems Modeling. Proceedings of the 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009*, Jgg. 29 of *Lecture Notes in Business Information Processing*, 2009.

[GLC+01] Fernand Gobet, Peter C.R. Lane, Steve Croker, Peter C-H. Cheng, Gary Jones, Iain Oliver and Julian M. Pine. Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, 5(6):236 – 243, 2001.

[GP96] T.R.G. Green and M. Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *J. Vis. Lang. Comput.*, 7(2):131174, 1996.

[GW04] Andrew Gemino and Yair Wand. A Framework for Empirical Evaluation of Conceptual Modeling Techniques. *Requirements Engineering*, 9(4):248–260, 2004.

[Int98] International Organization for Standardization. ISO 9241-11:1998 - Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability, 1998.

[Int06] International Organization for Standardization. ISO 9241-110:2006 Ergonomics of human-system interaction – Part 110: Dialogue principles., 2006.

[Kir02] Paul A. Kirschner. Cognitive load theory: implications of cognitive load theory on the design of learning. *Learning and Instruction*, 12(1):1–10, February 2002.

[KJL09] Sangeet Khemlani and P. N. Johnson-Laird. Disjunctive illusory inferences and how to eliminate them. *Memory & Cognition*, 37(5):615–623, 2009.

[KNS92] G. Keller, M. Nüttgens and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992.

[Lak92] George Lakoff. *Metaphor and Thought*, Kapitel The Contemporary Theory of Metaphor. Cambridge University Press, 1992.

[LS87] Jill H. Larkin and Herbert A. Simon. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11(1):65 – 100, 1987.

[May01] Richard E. Mayer. *Multimedia Learning*. Cambridge University Press, Cambridge, Massachusetts, 2001.

[Men07] Jan Mendling. *Detection and Prediction of Errors in EPC Business Process Models*. Phd thesis, Vienna University of Economics and Business Administration, 2007.

[Men08] Jan Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, Jgg. 6 of *Lecture Notes in Business Information Processing*. Springer, 2008.

[MH08] Daniel Moody and Jos Hillegersberg. *Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams*. Springer-Verlag, Berlin, Heidelberg, 2008.

[Mil56] G.A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[MNvdA07] Jan Mendling, Gustaf Neumann and Wil M. P. van der Aalst. Understanding the Occurrence of Errors in Process Models based on Metrics. In Robert Meersman and Zahir Tari, Hrsg., *On the Move to Meaningful Internet Systems 2007*, Jgg. 4803 of *Lecture Notes in Computer Science*, pp. 113–130. Springer, Vilamoura, Portugal, 2007.

[Moo05] Daniel Laurence Moody. Theoretical and Practical Issues in Evaluating the Quality of Conceptual Models: Current State and Future Directions. *Data & Knowledge Engineering*, 15(3):243–276, 2005.

[MP07] Ann Maes and Geert Poels. Evaluating Quality of Conceptual Modelling Scripts Based on User Perceptions. *Data & Knowledge Engineering*, 63(3):701–724, 2007.

[MR08] Jan Mendling and Jan Recker. Towards Systematic Usage of Labels and Icons in Business Process Models. In Terry Halpin, Hendrik Alexander Proper and John Krogstie, Hrsg., *12th International Workshop on Exploring Modeling Methods in Systems Analysis and Design*, CEUR Workshop Proceedings Series, pp. 1–13, Montpellier, France, 2008. CEUR.

[Mur89] Tadao Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[MvdA07] Jan Mendling and Wil M. P. van der Aalst. Formalization and Verification of EPCs with OR-Joins Based on State and Context. In John Krogstie, Andreas L. Opdahl and Guttorm Sindre, Hrsg., *Advanced Information Systems Engineering - CAiSE 2007*, Jgg. 4495 of *Lecture Notes in Computer Science*, pp. 439–453. Springer, Trondheim, Norway, 2007.

[NC99] J. C. Nordbotten and M. E. Crosby. The effect of graphic style on data model interpretation. *Information Systems Journal*, 9(2):139–155, 1999. 10.1046/j.1365-2575.1999.00052.x.

[ND86] Donald A. Norman and Stephen W. Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1986.

[Nie09] J. Nielsen. Usability 101: Introduction to Usability, 2009.

[NJL06] Mary R. Newsome and P. N. Johnson-Laird. How falsity dispels fallacies. *Thinking & Reasoning*, 12(2):214–234, 2006.

[OMG09a] OMG. Business Process Modeling Notation (BPMN). available at: http://www.omg.org/spec/BPMN/1.2/, January 2009. Version 1.2, formal/2009-01-03, The Object Management Group.

[OMG09b] OMG Unified Modeling Language (OMG UML): Superstructure. available at: http://www.omg.org/technology/documents/formal/uml.htm, February 2009. Version 2.2, formal/2009-02-02, The Object Management Group.

[Pai91] Allan Paivio. Dual Coding Theory: Retrospect and Current Status. *Canadian Journal of Psychology*, 45(3):255–287, 1991.

[RD07] Jan Recker and Alexander Dreiling. Does It Matter Which Process Modelling Language We Teach or Use? An Experimental Study on Understanding Process Modelling Languages without Formal Education. In Mark Toleman, Aileen Cater-Steel and Dave Roberts, Hrsg., *18th Australasian Conference on Information Systems*, pp. 356–366, Toowoomba, Australia, 2007. The University of Southern Queensland.

[RR07] Jan Recker and Michael Rosemann. Understanding the Process of Constructing Scales Inventories in the Process Modelling Domain. In Hubert Österle, Joachim Schelp and Robert Winter, Hrsg., *15th European Conference on Information Systems*, pp. 2014–2025, St. Gallen, Switzerland, 2007. University of St. Gallen.

[RRIG09] Jan Recker, Michael Rosemann, Marta Indulska and Peter Green. Business Process Modeling- A Comparative Analysis. *Journal of the Association for Information Systems*, 10(4), 2009.

[San77] J. L. Santa. Spatial transformations of words and pictures. *Journal of Experimental Psychology: Human Learning & Memory.*, 3:418–427, 1977.

[Sch00] A.W. Scheer. *ARIS - Business Process Modeling, 3rd Edition*. Springer Verlag, 2000.

[SR96] Mike Scaife and Yvonne Rogers. External cognition: how do graphical representations work? *Int. J. Hum.-Comput. Stud.*, 45(2):185–213, 1996.

[ST05] Keng Siau and Xin Tan. Improving the quality of conceptual modeling using cognitive mapping techniques. *Data Knowl. Eng.*, 55(3):343–365, 2005.

[Swe88] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science: A Multidisciplinary Journal*, 12(2):257–285, 1988.

[Ves91] I. Vessey. Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature*. *Decision Sciences*, 22(2):219–240, 1991.

[Wer38] M. Wertheimer. *A sourcebook of Gestalt psychology*, Kapitel Laws of organization in perceptual forms, pp. 7188. Routledge and Kegan Paul, London, UK, 1938. Reprinted from Psychologische Forschung, 4, pp. 301350, 1923.

[YR95] Y. Wand and R. Weber. On the deep structure of information systems. *Information Systems Journal*, 5(3):203–223, 1995. 10.1111/j.1365-2575.1995.tb00108.x.