Imperative versus Declarative Process Modeling Languages: An Empirical Investigation

Paul Pichler[1], Barbara Weber[1], Stefan Zugal[1], Jakob Pinggera[1], Jan Mendling[2], and Hajo A. Reijers[3]

[1] University of Innsbruck, Austria
paul.pichler@student.uibk.ac.at
{barbara.weber,stefan.zugal,jakob.pinggera}@uibk.ac.at
[2] Humboldt-Universität zu Berlin, Germany
jan.mendling@wiwi.hu-berlin.de
[3] Eindhoven University of Technology, The Netherlands
h.a.reijers@tue.nl

Abstract. Streams of research are emerging that emphasize the advantages of using declarative process modeling languages over more traditional, imperative approaches. In particular, the declarative modeling approach is known for its ability to cope with the limited flexibility of the imperative approach. However, there is still not much empirical insight into the actual strengths and the applicability of each modeling paradigm. In this paper, we investigate in an experimental setting if either the imperative or the declarative process modeling approach is superior with respect to process model understanding. Even when task types are considered that should better match one or the other, our study finds that imperative process modeling languages appear to be connected with better understanding.

Keywords: Imperative and Declarative Business Process Models, Cognitive Dimensions Framework, Empirical Research.

1 Introduction

At the present stage, formal properties of business process models like liveness and boundedness are quite well understood [1]. In contrast to these aspects, we know rather little about theoretical foundations that might support the superiority of one process modeling language in comparison to another one. There are several reasons why suitable theories are not yet in place for language design, most notably because the discipline is still rather young. Only little research has been conducted empirically in this area so far, e.g., [2,?] which relate model understanding to the modeling language and to model complexity.

The lack of empirical research on language quality has contributed to a notable, continuous invention of new techniques and to the claims on the supposed superiority. For instance, Nigam and Caswell introduce the OpS technique in which "*the operational model is targeted at a business user and yet retains the* *formality needed for reasoning and, where applicable, automated implementation*" implying that existing languages fall short on these characteristics [3, p. 429]. In a Poplin white paper, Owen and Raj claim a general superiority of one modeling language over another, i.e., BPMN over UML Activity Diagrams, because "*[BPMN] offers a process flow modeling technique that is more conducive to the way business analysts model*" [4, p.4]. As a final example, Smith and Fingar state in their book that "*BPML is the language of choice for formalizing the expression, and execution, of collaborative interfaces*" [5, p.205]. We do not want to judge on the correctness of these statements here. Rather, we wish to emphasize that a clear and objective baseline to judge such claims is in demand.

The matter of understanding is well-suited to serve as a pillar for discussing process modeling language quality. Insights from cognitive research on programming languages point to the fact that 'design is redesign' [6]: A computer program is not written sequentially; a programmer typically works on different chunks of the problem in an opportunistic order which requires a constant reinspection and comprehension of the current work context. If we assume that process modelers design their models in a similar fashion, we clearly have to accept the importance of *understanding* as a quality factor. In other words, characteristics of a process modeling language presumably facilitate comprehension to differing degrees in a particular context.

To investigate whether process modeling languages actually offer different levels of support for sense-making, we will necessarily need to limit our scope. One of the important watersheds that exists between process modeling languages is the one between *imperative* and *declarative* process modeling languages. For the recently developed ConDec, a declarative process modeling language, its first design criterion has been that "*the process models developed in the language must be understandable for end-users*" [7, p.15]. While it is claimed in the same work that imperative languages, in comparison, deliver larger and more complex process models, only anecdotic evidence is presented to support this. Also, in the practitioner community opinions are manifold about the advantages of declarative and imperative languages to capture business processes, see for example [8,**?**,**?**]. These claims and discussions clearly point to the need for an objective, empirically founded validation of the presumed advantages of the different types of process modeling languages, which motivates the scope of our research.

The contribution of this paper is that it empirically examines if either imperative or declarative process models are superior with respect to understanding matters. To this purpose, we will test a set of theoretically grounded propositions about the differences between the imperative and declarative process modeling approach. The paper is structured as follows. Sect. 2 provides the background for our research. Sect. 3 describes the experimental definition and planning, covering the experimental setup and design. Sect. 4 presents the experimental execution and the analysis of collected data. Furthermore, the results are discussed in this section. Sect. 5 concludes the paper by providing a summary and an outlook.

2 Background

The differentiation between imperative and declarative languages has its roots in computer programming. Imperative programming implies to "*say how to do something*" [9, p.406], whereas declarative programming implies to "*say what is required and let the system determine how to achieve it*" [9, p.406]. Similar to imperative programming, imperative process modeling is characterized by a so-called 'inside-to-outside' approach. It primarily specifies the procedure of how work has to be done. Simply put, imperative modeling languages require all execution alternatives to be explicitly specified in the model before the execution of the process. All new alternatives must be added to the model during build-time. It is argued that this results in process models being over-specified [7]. Declarative process modeling, by contrast, is referred to as an 'outside-to-inside' approach. In contrast to imperative languages, declarative languages do not specify the procedure *a priori*. Instead of determining how the process has to work exactly, only its essential characteristics are described. Adding new constraints to the model limits the number of execution alternatives [7]. This may be understood as follows: Initially, only the process activities are in the model, allowing every possible execution behavior. By adding constraints to the model, execution alternatives are discarded step by step. Figure 1 shows an example of a declarative process model consisting of three activities A, B, and C and two constraints. The constraint attached to activity C specifies that it has to be executed at least once. The constraint between activities A and B requires that the execution of activity B is preceded by activity A. Except for these restrictions, the activities in the model can be executed arbitrarily often and in any order.

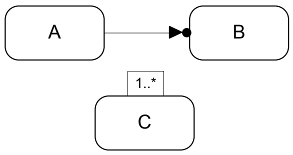

Fig. 1. Declarative Process Model

Clearly, the above mentioned claims need to be substantiated in terms of appropriate theories. The Cognitive Dimensions Framework (CDF) offers a reference for discussing and evaluating various types of notations based on their cognitive effectiveness [10]. Its development is based on the 'mental operations theory' [6], which in essence states that a notation performs better if fewer mental operations are required to perform a task. In this way, a "matched pair" between particular, notational characteristics and a specific task gives the best performance. This view has evolved and matured over the years towards the CDF [10,?], which contains many different characteristics to distinguish notations from each other. In particular, the dimensions *hard mental operations* (to understand a model) and *hidden dependencies* (between notation elements) directly apply to process modeling understanding [10].

In line with the CDF, different notations should be judged *relatively*, i.e., in terms of their aptitude towards different types of understanding tasks. "*A notation is never absolutely good, therefore, but good only in relation to certain tasks*" [10, p.3]. In this vein, it seems appropriate to investigate whether imperative languages are better understandable with tasks containing a particular type of information and declarative languages with tasks containing another type of information [10]. For such a distinction, the classification between sequential and circumstantial information is relevant.

Sequential information explains how input conditions lead to a certain outcome. An example of a statement containing sequential information is: "Activity X must be directly preceded by activity Y". As this example demonstrates, sequential information often concentrates on what actions could be either next or previous in a model [6]. In other words, sequential information typically relates to actions immediately *leading to* or *following from* a certain outcome. On the other hand, *circumstantial information*, given an outcome, relates to the overall conditions that produced that outcome. An example of a statement containing circumstantial information is: "If activity X or Y has been executed, it is possible to terminate a process instance by executing at least one additional activity". As this example demonstrates, circumstantial information frequently corresponds to what (combination of) circumstances will cause a particular outcome or action [6]. In this context, circumstantial information typically relates to conditions that *have* or *have not* occurred.

Empirical evidence has already been established that imperative programming languages display sequential information in a readily-used form, while declarative languages display circumstantial information in a readily-used form [6]. Based on the similarities between software programs and process models, it may be assumed, therefore, that a similar, relativist viewpoint could also provide a theoretical basis for the comparison of imperative versus declarative process modeling languages [11]. Consequently, the following set of propositions may be advanced, which are in line with those proposed in an earlier paper [11]:

P1. Given two semantically equivalent process models, establishing sequential information will be easier on the basis of the model that is created with the process modeling language that is relatively more imperative in nature.

P2. Given two semantically equivalent process models, establishing circumstantial information will be easier on the basis of the model that is created with the process modeling language that is relatively more declarative in nature.

To test these expectations, we will next describe an experimental design for that purpose.

3 Experimental Definition and Planning

This section introduces the hypotheses, describes the subjects, objects, factors, factor levels and response variables of our experiment. It will also present the instrumentation, data collection procedure, and experimental design. Finally, the parameters we controlled for in our experiment are discussed.

Factor and Factor Levels. The considered factors were *model type* and *task type*, with two factor levels each. For the model type factor, we considered the factor levels of *imperative* and *declarative*. For the task type factor, we considered the factor levels *sequential* versus *circumstantial*.

Subjects. Students enrolled in classes on business process management were participating as subjects in the experiment.

Objects. In preparation for the experiment, four semantically equivalent process model pairs were created[1]. Semantic equivalence was ensured by testing valid traces based on both model variants. BPMN was used to create the imperative models, and ConDec to create the declarative models. Both imperative and declarative process models were created considering the following criteria: 1) Correctness, 2) Executability and 3) Representativeness. *Correctness* is the precondition of executability, a characteristic of understandable process models defined by the SEQUAL Framework [12]. For imperative models soundness and structuredness were considered as correctness notions [13]. For declarative models, in turn, absence of dead activities and conflicts was required [7]. To ensure *executability* we transformed the imperative models to Petri nets allowing us to apply the token game. For declarative models, in turn, we tested executability using the in-built verification functionality of DECLARE [7]. To ensure content validity, i.e., the *representativeness* of the experimental objects, we ensured that the four model pairs covered the core concepts of both the imperative and declarative paradigm. Imperative process models covered the five basic control flow patterns (i.e., sequence, exclusive choice, simple merge, parallel split and synchronization) [14] as well as loops. Declarative models, in turn, covered all major constraint groups (i.e., existence, relation and negation constraints [15]).

Tasks. For each of the model pairs, i.e., a declarative versus an imperative model, four sequential and four circumstantial tasks had to be created (comprising understandability questions) considering the following criteria: 1) Typical constructs, 2) Model parts, 3) Difficulty and 4) Consistency. To maintain content validity it had to be ensured that the experimental tasks cover all relevant aspects of understandability for each modeling language. In [16], the four constructs order, concurrency, exclusiveness, and repetition are mentioned as being crucial for the understanding of imperative process models and were therefore considered for creating the sequential tasks. Circumstantial tasks, in turn, were adjusted in terms of the constraints groups which determine declarative representativeness (i.e., existence, relation, and negation constraints).

Sequential information usually affects local parts of a process model [17]. Consequently, sequential tasks were formulated with reference to local actions (e.g., next or previous) in the model. Contrary to sequential information,

[1] The material used for this study can be downloaded from:
`http://barbaraweber.org/experiments/2010_Declarative_vs_Imperative.pdf`

circumstantial information tends to affect the process model rather globally [17]. Therefore, circumstantial tasks were formulated such that they ask for (the combination of) circumstances that caused or will cause a particular action within the process. To establish a balanced level of difficulty among the experimental tasks, and hence avoid that tasks which are either too easy or too difficult impact the result of the experiment, a pre-test was conducted. To ensure that only the information captured by the tasks is of relevance for the experimental outcome, the tasks were formulated consistently in respect of their structure and the use of terms.

Response Variables. To compare declarative and imperative modeling languages we defined the following response variables: 1) *accuracy* as measured by the number of correctly answered questions (tasks) and 2) *speed* by measuring the time needed to complete the tasks. Since four sequential and four circumstantial tasks had to be completed for each model pair, accuracy values could range between 0 and 4 for sequential and circumstantial tasks respectively.

Hypotheses. A statistical test with two factors is always associated with three null hypotheses [18], one for each factor and one for the interaction between the factors:

– ***Null Hypothesis*** H_1***:*** *There is no significant difference in the performance (in terms of accuracy and speed) of imperative and declarative models.*
– ***Null Hypothesis*** H_2***:*** *There is no significant difference in the performance (in terms of accuracy and speed) of sequential and circumstantial tasks.*
– ***Null Hypothesis*** H_3***:*** *There is no significant interaction between the factor model type and the factor task type (in terms of accuracy and speed).*

Parameters. In addition to the described factors other variables can affect the response variables under examination and therefore need to be controlled [19]. For this experiment we considered three main parameters that can influence the understandability of process models, i.e., model characteristics, domain knowledge and personal factors [20]. *Model characteristics* we controlled involved visual layout and structural attributes. To control *visual layout* we maximized symmetry, minimized bends and minimized edge crosses for both model variants (i.e., imperative and declarative), since they are known factors which influence model understandability [21]. To control the *structural attribute* size, which has significant impact on model understandability [22], we had to ensure that the imperative and declarative variants have equal size. Consequently, to avoid size rather than the used modeling paradigm dictating the outcome, the used model pairs comprised two small and two large models for each factor level. Fig.2 shows an experimental model pair consisting of a small imperative and a small declarative model.

To eliminate the influence of *domain knowledge*, activities of the process models were labeled with random letters. To control *personal factors*, the selection of experimental subjects comprised preferably persons with uniform knowledge regarding business process modeling. Nevertheless, the subjects had to complete

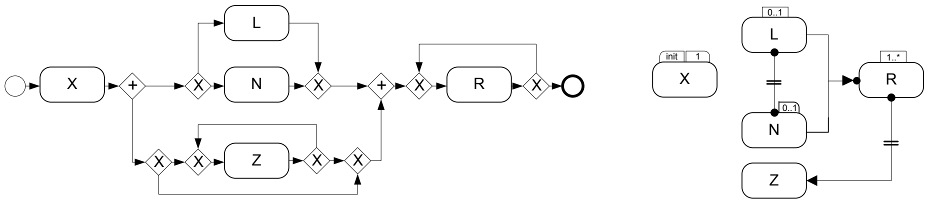

Fig. 2. Imperat. BPMN model (left) and equivalent, declarat. ConDec model (right).

a questionnaire at the beginning of the experiment allowing us to analyze how possible variations between the individual modeling knowledge influenced the model understanding and quantify the distribution of modeling experience between the imperative and declarative modeling approach.

Experimental Design. The experimental design is based on the guidelines for designing experiments from [19]. Following these guidelines, a *randomized 2x2 factorial experiment* has been designed, which investigates the influence of two factors with two factor levels each. The experiment is called *randomized*, since subjects are assigned to groups randomly. Fig. 3 illustrates the described setup: Overall, the experiment comprised four model pairs. Depending on their group, subjects either started with the imperative variant of Model Pair 1 or with the semantically equivalent declarative variant. For two of the remaining model pairs, the levels of factor model type were switched for the two groups, i.e., overall, every subject worked on two declarative and two imperative process models. Regarding factor task type every subject worked on both factor levels (i.e., sequential and circumstantial tasks) for each model pair. To ensure independence of samples, both sequential and circumstantial tasks were presented in a random, and thus unique order to each subject.

Instrumentation and Data Collection Procedure. The participants conducted the experiment using the Cheetah Experimental Platform [23], which guided them through the experiment. The tool also automatically logged the given answers, as well as the time that was needed to accomplish the experimental tasks.

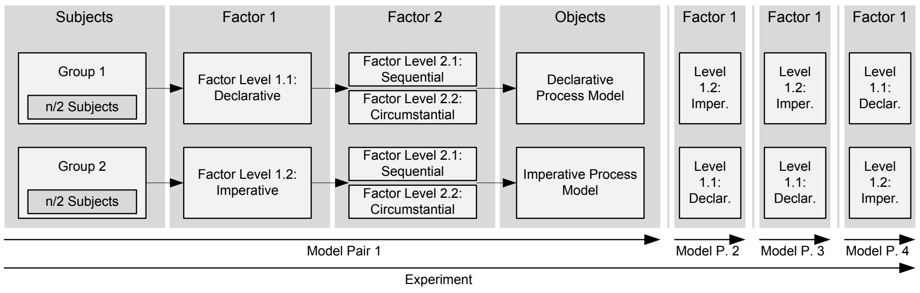

Fig. 3. Experimental Design

4 Performing the Experiment

This section deals with the experiment's execution. Sect. 4.1 covers operational aspects, i.e., how the experiment has been executed. Then, in Sect. 4.2 data is analyzed and subsequently discussed in Sect. 4.3.

4.1 Experimental Operation

Experimental Preparation. Four semantically equivalent model pairs were created for the empirical test. Additionally, for each model pair a set of four sequential and four circumstantial tasks was developed (cf. Sect. 3). To ensure the overall understandability of the experimental setup and a balanced degree of difficulty, a pre-test was conducted.

Experimental Execution. The experiment was conducted in July 2010. In sum, 28 subjects from the Humboldt Universität zu Berlin and the University of Innsbruck participated in this empirical test. Subjects had one week time to complete the experiment in an "offline" mode, i.e., they were not constantly monitored.

Fig. 4 shows the structure of the experiment by means of a complete run: Each student received a PDF file containing the introduction, hints for the experiment and troubleshooting as well as specific instructions on how to download and execute it.[2] Having downloaded the experiment, the subjects had to complete the questionnaire about their personal modeling knowledge and experience. An example followed providing the solution as well as the respective explanation to every test task. During the experimental phase, the subjects had to complete eight tasks (four sequential and four circumstantial ones) for each of the four models. A legend with the used modeling elements was attached to every process model.

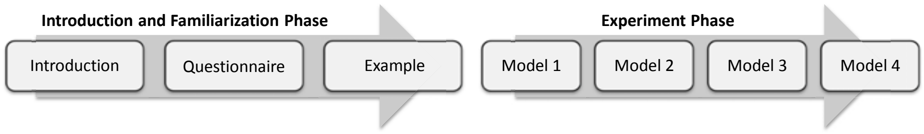

Fig. 4. The Structure of the Experiment

Data Validation. Once the experimental study was carried out, the logged data were analyzed regarding their consistency and plausibility. Finally, data provided by 27 students were used in our data analysis. Data from one student had to be removed because it was incomplete.

[2] The version of CEP which was used for the experiment including the experimental workflow can be downloaded from:
`http://barbaraweber.org/experiments/2010_Declarative_vs_Imperative.zip`

4.2 Data Analysis

In the following we describe the analysis and interpretation of data.

Descriptive Analysis. To give an overview of the experiment's data, Table 1 shows mean values and standard deviation for accuracy and speed.

Table 1. Descriptive Statistics

Model Type	Task Type	N	Accuracy Mean(score)	Accuracy Std. Dev.	Speed Mean(min.)	Speed Std. Dev.
Declarative	Sequential	54	2.61	1.14	2.21[3]	1.13
	Circumst.	54	2.44	1.18	2.79	1.69
Imperative	Sequential	54	3.26	0.89	1.91	0.98
	Circumst.	54	2.87	0.95	2.06	0.80
Declarative		108	2.53	1.16	2.50	1.46
Imperative		108	3.06	0.94	1.98	0.89
	Sequential	108	2.94	1.07	2.06	1.06
	Circumst.	108	2.66	1.10	2.43	1.37

Hypotheses Testing. In a next step, the hypotheses introduced in Sect. 3 were tested. The empirical test in this work was designed as a two-way factorial experiment. Accordingly, a two-way factorial research design requires a statistical analysis method that allows the interpretation of both factors together (i.e., (M)ANOVA). Since the requirements for the application of ANOVA were not satisfied, we applied the Sheirer-Ray-Hare test, a non-parametric alternative for ANOVA [18].

First we discuss the result of testing null hypothesis H_3, since the effect of a factor can be interpreted individually only when there is no evidence that it interacts with another factor [19].

Null Hypothesis H_3: With an obtained p-value (=significance value) of 0.45 (>0.05), null hypothesis H_3 cannot be rejected at a confidence level of 95% for the response variable of accuracy. Also, the results for the response variable of speed turned out to be insignificant (p-value of 0.37, >0.05). Hence, there is no statistically significant interaction between the factors model type and task type. Since there is no evidence of a significant interaction, the effect of the factors can be interpreted individually [19].

Null Hypothesis H_1: With an obtained p-value of 0.001 (<0.05), null hypothesis H_1 has to be rejected for the response variable *accuracy* at a confidence level of 95%. Hence, there is a statistically significant difference between the alternatives of the factor model type: *Imperative models have a better performance in terms of accuracy than declarative models, regardless of the factor task type.* With an obtained

[3] A lower number in terms of response variable "speed" implies a better result.

p-value of 0.002 (<0.05), null hypothesis H_1 has to be rejected for response variable *speed*. This result indicates that *imperative models have a better performance in terms of speed than declarative models, regardless of the factor task type.*

Null Hypothesis H_2*:* With an obtained p-value of 0.06, which just exceeds the 0.05 level, null hypothesis H_2 cannot be rejected at a confidence level of 95%, *i.e., there is no statistically significant difference between the alternatives of the factor task type, i.e., sequential or circumstantial tasks, in terms of accuracy.* Since there is the possibility that one type of process models is significantly better performing with one type of the tasks, and the other type of process models is not, we additionally analyzed this hypothesis separately for each factor level of factor model type using the Mann-Whitney-U test. With an obtained p-value of 0.02 (<0.05), a statistically significant difference between the alternatives of the factor task type could be established when imperative models are used, i.e., *sequential tasks compared to circumstantial tasks lead to a better performance in terms of accuracy when imperative models are used.* For declarative models, in turn, with a p-value of 0.51 >0.05 no statistically significant results could be obtained. For the response variable of speed, null hypothesis H_2 has to be rejected (p-value of 0.01, <0.05): *Sequential tasks compared to circumstantial tasks lead to a better performance in terms of speed than circumstantial tasks, regardless of the factor model type.*

4.3 Discussion

The objective of this paper has been to investigate if either the imperative or the declarative process modeling approach is superior with respect to understanding matters. The set-up for this investigation has been grounded on insights from cognitive research on programming languages. Our findings suggest that imperative process models are significantly better understandable than declarative models, irrespective of the type of tasks involved (sequential vs. circumstantial).

This result, however, must be treated with care, since an ex-post analysis of the process modeling experience of our subjects revealed that the experimental subjects were rather familiar with imperative process modeling, but at best only to a limited extent to declarative modeling. Based on this imbalance, a subsequent analysis was examined. This revealed that a learning effect for declarative models might have occurred during the experiment. Since this affects generalizability of the results, replications regarding this research objective are required with subjects having a more balanced level of familiarity for both modeling paradigms.

In addition to examining imperative versus declarative process models, our goal has been to test if the theoretical axioms of the CDF, which were originally established for computer programming as part of extensive cognitive research, also apply to business process modeling. Based on the obtained data it could only be confirmed that tasks containing sequential information are better understandable using imperative process models, but not that tasks containing circumstantial information are better understandable using declarative models. Regarding the response variable *accuracy*, sequential tasks were easier to understand using imperative models. However, we could not confirm that circumstantial tasks are easier to understand using declarative models. In terms of the response variable *speed*, sequential tasks turned out to be better performing irrespective of the type of model concerned.

We effectively conducted the experiment with a homogeneous group of students. Still, further potential limitations must be considered. In particular, the relatively low number of experimental subjects constitutes a limitation as tests converge towards significant results with more subjects. Moreover, due to the small number of participants in the pre-test, it was not possible to conduct a statistically significant reliability analysis on the internal consistency of the different understandability tasks. Even though we tried to balance the level of difficulty with a pre-test, it cannot be entirely excluded that this issue might have influenced the experimental result. Another limitation regarding the generalizatbility of our results relates to the fact that our experiment only compares one concrete modeling language representing each process modeling approach.

5 Summary and Outlook

In this paper, we compared the imperative process modeling approach with the declarative approach with reference to understanding based on insights from cognitive research on programming. Essentially, imperative process models turned out to be more comprehensible than declarative process models, irrespective of the type of task involved. However, based on the imbalance of subjects' familiarity with imperative and declarative process modeling, this result must be treated with care.

A further insight concerns the theoretical axioms of the Cognitive Dimensions Framework, stating that tasks containing sequential information are better understandable using imperative languages, and tasks containing circumstantial information are better understandable using declarative languages. This could be confirmed partially. Apparently, sequential tasks are better understandable, regardless whether an imperative or declarative process model was used.

The most important direction for future research we identify would be to replicate the experiment in a situation where the participants' knowledge of and experience with both imperative and declarative languages is less skewed. One can argue that this may be hard to establish, given the dominant emphasis in many settings on the use of imperative approaches, for example in academic programs. A step forward here may be taken by involving people with no training or background at all in process modeling, who can be provided equal amounts of training time in both paradigms.

References

1. Reisig, W., Rozenberg, G. (eds.): APN 1998. LNCS, vol. 1491. Springer, Heidelberg (1998)
2. Recker, J., Dreiling, A.: Does It Matter Which Process Modeling Language We Teach or Use? An Experimental Study on Understanding Process Modelling Languages without Formal Education. In: Proc. ACIS 2007, pp. 356–366 (2007)

3. Nigam, A., Caswell, N.: Business artifacts: An approach to operational specification. IBM Systems Journal 42(3), 428–445 (2003)
4. Owen, M., Raj, J.: BPMN and Business Process Management: Introduction to the New Business Process Modeling Standard. Popkin, Technical report (2003), `http://whitepaper.talentum.com/whitepaper/view.do?id=7050`
5. Smith, H., Fingar, P.: Business Process Management: The Third Wave. Meghan-Kiffer Press (2003)
6. Gilmore, D.J., Green, T.R.: Comprehension and recall of miniature programs. IJMMS 21, 31–48 (1984)
7. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, TU Eindhoven (2008)
8. Korhonen, J.: Evolution of agile enterprise architecture (April 2006), `http://blog.jannekorhonen.fi/?p=11` (retrieved May 10, 2011)
9. Roy, P.V., Haridi, S.: Concepts, Techniques, and Models of Computer Programming. The MIT Press (2004)
10. Green, T.R.: Cognitive dimensions of notations. In: Proc. BCSHCI 1989, pp. 443–460 (1989)
11. Fahland, D., Mendling, J., Reijers, H.A., Weber, B., Weidlich, M., Zugal, S.: Declarative Versus Imperative Process Modeling Languages: The Issue of Understandability. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) EMMSAD 2009. LNBIP, vol. 29, pp. 353–366. Springer, Heidelberg (2009)
12. Krogstie, J., Sindre, G., Jørgensen, H.: Process models representing knowledge for action: a revised quality framework. EJIS 15, 91–102 (2006)
13. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. Springer, Heidelberg (2008)
14. Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P., Mulyar, N.: Workflow Control-Flow Patterns. A Revised View. BPM Center Report, 6–22 (2006)
15. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Tennenholtz, M. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
16. Melcher, J., Mendling, J., Reijers, H.A., Seese, D.: On Measuring the Understandability of Process Models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 465–476. Springer, Heidelberg (2010)
17. Weidlich, M., Zugal, S., Pinggera, J., Fahland, D., Weber, B., Reijers, H.A., Mendling, J.: The Impact of Sequential and Circumstantial Changes on Process Models. In: Proc. ER-POIS 2010, pp. 43–54 (2010)
18. Dytham, C.: Choosing and Using Statistics. A Biologist's Guide. John Wiley & Sons (2003)
19. Juristo, N., Moreno, A.M.: Basics of Software Engineering Experimentation. Kluwer Academic Publishers (2001)
20. Mendling, J., Strembeck, M.: Influence factors of understanding business process models. In: Proc. BIS 2008, pp. 142–153 (2008)
21. Purchase, H.: Which Aesthetic has the Greatest Effect on Human Understanding? In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997)
22. Mendling, J., Reijers, H.A., Cardoso, J.: What Makes Process Models Understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 48–63. Springer, Heidelberg (2007)
23. Pinggera, J., Zugal, S., Weber, B.: Investigating the process of process modeling with cheetah experimental platform. In: Proc. ER-POIS 2010, pp. 13–18 (2010)