Towards Automated Support for Case Management Processes with Declarative Configurable Specifications

Irina Rychkova

Centre de Recherches en Informatique, University Paris 1 Pantheon-Sorbonne,
90, rue Tolbiac, 75013 Paris, France
Irina.Rychkova@univ-paris1.fr

Abstract. Until recently, efficiency gained through process automation and control was the main preoccupation of BPM practitioners. As a result, the majority of mainstream process modeling standards today is characterized by the imperative modeling style. This style encourages a modeler to commit to a well-determined process execution scenario already at the early design stages. For *case management processes*, however, a strict commitment to a predefined control flow is considered by organizations as a serious handicap. This is the main reason why case management as well as other knowledge-intensive processes in the organizations mostly remain "pen and paper". In this article we demonstrate how *configurable data objects and context-based configuration rules* can be integrated into a process model in order to improve the process post-design adaptability and to pave the road for case management automated support. These concepts are defined as a part of DeCo (the Declarative Configurable process specification language). DeCo is a declarative modeling approach that is currently under development. We illustrate our results on the example.

Keywords: Business process modeling, BPMN, declarative modeling, configurability.

1 Introduction

During the last decades, business process management (BPM) became an imperative for efficient functioning and evolution of organizations and gave a rise to the third wave of research interest in business process modeling and analysis [1]. Since then, the main efforts of researchers have been focused on development of process modeling languages that would be easy to understand by both technical and business users and that would provide better control over processes. As a result, the majority of the process modeling methodologies widely used today (e.g. BPMN, UML, EPC) is characterized by a powerful graphical notation, a rich design environment, and by the *imperative* style of their models.

Imperative modeling largely contributes to the process control [2]. Modeling approaches based, for example, on coupling a graphical language such as BPMN and an operational language such as BPEL infinitely attract BPM practitioners giving them a toolbox to design, execute, test and eventually instantiate and deploy the process models.

M. La Rosa and P. Soffer (Eds.): BPM 2012 Workshops, LNBIP 132, pp. 65–76, 2012.

Requiring large upfront investments into a process scenario definition, such approaches still pay off very well for deterministic, repetitive processes (such as automated production lines). Assuming that execution scenarios are changing rarely, once the process is automated, an organization will quickly benefit from the economy of scales.

Latest publications show the increasing interest of BPM practitioners in unstructured, knowledge-driven processes. The term "knowledge-driven" or "knowledge-intensive" refers to the fact that a process execution scenario significantly depends on knowledge of a human expert rather then on the predefined process model. Such process can vary from one execution to another demonstrating large unpredictability [3][4][5]. One of the examples of knowledge-driven processes is case management (CMP). OMG defines case management as *"A coordinative and goal-oriented discipline, to handle cases from opening to closure, interactively between persons involved with the subject of the case and a case manager or case team"*.

Seeking to increase their efficiency by process automation, organizations also admit that for the knowledge-driven processes such as CMP, the ability to adapt the process scenario according to a situation (we call it post-design adaptability) is the most essential. This makes automation of case management processes following an imperative approach too expensive: a number of modifications to initial process model will quickly outweigh any automation benefits [6].

In the CMPM request for proposal released in November 2009 [4], OMG formulates in detail the problem of case management process modeling and support, illustrating the need for another paradigm.

Declarative Configurable specification language (DeCo) was first introduced in [7]. With DeCo, we exploit an idea that an activity of "scenario definition" for a business process is not limited to a process design time (as imperative BPM requires) but it makes an integrated part of process deployment and even execution. In this paper, we define configurable process elements (namely, configurable data objects) and context-based configuration rules for DeCo. These modeling concepts allow one to continuously adapt the process along its lifecycle and, consequently, they pave the road for the automated decision-making support of knowledge workers. We illustrate our ideas on the example of a mortgage approval process.

The reminder of this article is organized as follows: in Section 2 we discuss the related works. In Section 3 we introduce the mortgage approval process (a CMP example), present our motivations in creating DeCo and relate this work with our previous research. In Section 4 we discuss the modeling principles of DeCo and introduce the configurable data object and context-based configuration rules concepts. In Section 5 we present conclusions and discuss our future work.

2 Related Works

Today, the majority of de-facto modeling standards including UML, EPC, BPMN exploit the *imperative* modeling style. Imperative process models are suitable for simulation [9], thus they can be highly advantageous for practitioners helping them to control the process and to exclude incorrect or undesirable scenarios already at design. BPEL[10] is a standard executable language for process models documented

in BPMN. Operational semantics for process model execution based on Petri Nets and Pi calculus is considered in [11][12]. The research reported in [13] proposes a formal semantics of BPMN defined using Petri nets. In [14], the technique for simulation and analysis of process specified with DEMO [15] is presented.

Yet providing the means for simulation and control, imperative process models are proven to be restrictive [16]: specification of numerous options and variations for the sake of process adaptability becomes difficult if at all possible in such models [17][5]. Alternatively, a process can be specified *declaratively*: this modeling style supports non-determinism and allows modeler to postpone the decision making about the process execution scenario until its deployment or even execution. In [18], the detailed comparison of imperative and declarative modeling styles is provided. In [19][20], SEAM (for Systemic Enterprise Architecture Methodology) is presented. In SEAM, processes in an organization can be modeled as *joint actions* with implicit (nondeterministic) execution scenarios. Modeler refines SEAM models selecting an appropriate level of details. Thus the detailed specification of the process flow can be postponed. In the similar way, the MAP methodology [21][22] addresses the process specifications using the notions of *intentions* and *strategies*. Each intention in MAP can be realized following one or multiple alternative strategies, leaving the process execution largely nondeterministic and, therefore, adaptable for a given context. MAP notation was used to address process variability and to model process lines in [37].

Whereas some researchers decide to develop their modeling notations from scratch, others extend the existing standards providing them with the desired properties. In [23] both the EPC (Event Process Chain) and the BPMN metamodels are extended with elements for modeling process goals and process performance measures. In [24] the BPMN notation is extended to provide the concepts for querying the business process definitions and extracting the business process patterns. In [25][26] the (imperative) EPC notation is extended with the concepts for process configurability along the control-flow, data, and resource perspectives.

While supporting process adaptability, declarative models encompass a significant ambiguity and, as a consequence, are not suitable for simulation. Numerous efforts to achieve adaptability and control under the roof of the same process modeling approaches have been reported in the literature. The underlying semantics is ranged from LTL (linear temporal logic) and FOL (first-order logic) to Petri nets enabling automated modelchecking and theorem proving techniques for model validation and analysis known from the software engineering. In [27], DecSerFlow language for Declarative modeling of service flows is presented. In [28] the same authors present the DECLARE system for supporting declarative (loosely-structured) process models. The formal semantics of DECLARE and DecSerFlow is based on LTL. The formal semantics for the Configurable Integrated EPC (C-iEPC) modeling notation presented in [25] is based on FOL and serves to validate the process model correctness.

3 DeCo: Motivation and Relation to Our Previous Research

In [29][30] the declarative semantics for a graphical modeling notation for enterprise modeling called SEAM is discussed. In [29] we consider the variability aspects of

business process modeling and propose declarative modeling approach and semantics based on Alloy [31]. Alloy specification language allows one to validate the conformance between a high-level declarative design specification of a process with its low level imperative implementation specification.

In [32][7], we put the research of the previous years together in order to develop an approach offering to a modeler an extensive configurability opportunities while implementing the principles of declarative modeling and supporting automated model analysis and step-wise refinement [33]. We called this approach DeCo – for Declarative Configurable process specifications.

In [36] we introduce a configurable roles and add this concept to MAP notation.

This work introduces another two modeling concepts of DeCo: configurable data objects and contextual configuration rules. We illustrate the use of these concepts on the example of mortgage approval process.

3.1 Motivating Example: Mortgage Approval Process

Mortgage approval process is a typical example of a case management process. In this paper, we provide a generic mortgage approval process description as defined by different financial institutions in the USA. The information provided below represents a compilation of guidelines and descriptions of mortgage approval process, provided by different loan consulting firms, financial advisors, and banks and available on the web (e.g. http://www.homebuyinginstitute.com/, http://www.mortgage-resource-center.com/, http://homebuyereducation.bankofamerica.com/, etc.)

A mortgage is a loan for buying a house. The mortgage approval process can be divided into the following sub processes: Pre-qualification, Formal Application, Document Review, Pre-approval, Property Appraisal, Final Approval, Closing. Whereas the order of these main sub processes varies rarely, the documents required, the sequences of tasks, the participants of each sub process can be different depending on the place, the financial institution's policies, and the applicant's situation and requirements. In this paper, we focus only on the Formal Application sub process:

Formal Application

0 An applicant can request the application package by e-mail or by post. Alternatively, all the forms can be downloaded from the web site of a prospective lender.

1 A mortgage application can be submitted electronically or during a personal meeting with the mortgage lender.

2 The exact set of documents may vary depending on the financial institution and the particular situation of an applicant. These documents may include: the social security card, record for past two years for residence address, employer data, various Internal Revenue Service (IRS) forms, recent pay-stubs, etc.

3-5 During the formal application, in some states, the mortgage lender provides the applicant with a Good Faith Estimate (GFE) of costs of loan closing; the applicant can be also asked to make a final decision on the type of mortgage loan and to lock in an interest rate.

6 Usually during the formal application submission an applicant has to pay the application fee and the appraisal fee. For some agencies, however, the appraisal will be charged later, whereas the application procedure can be free of charge.

Though substantially simplified, the description above illustrates the variability of activities, actors, and information involved into the process. The process scenario can also differ substantially depending on the execution context (e.g. the country, state, agency). For a financial organization operating globally and dealing with multiple environments definition of a single process model becomes a challenging task.

In our previous work reported in [32] we modeled this Formal Application sub process using BPMN and formulated *five* modeling challenges common for imperative modeling approaches in general:

1. Need to specify task inputs/outputs while distinguishing obligatory and optional data objects, alternatives (possible replacements), and synonyms (identical artifacts called differently).
2. Need to specify role hierarchy, alternative roles and synonyms.
3. Need to specify optional, obligatory, alternative task and synonyms.
4. Need to specify multiple control flow possibilities.
5. Need to specify an impact of data on different tasks and the task flow.

4 Improving Adaptability of Mortgage Approval Process Model with DeCo

4.1 Replacing Imperative Scenario with Declarative Specification

The graphical notation of DeCo is based on BPMN [34] whose modeling concepts are widely used and recognized by practitioners. But similarities terminate here since, compared to BPMN, DeCo implements the declarative modeling principles. These principles allow one to postpone the decision making about the eventual process scenario until its deployment or even execution.

Declarative approach to process modeling represents an alternative to continuous exception definition, use of "if-then-else" or "switch" constructions within a traditional, workflow-based process model [31]. Instead of modeling a flow of process activities, in DeCo, we focus on modeling individual activities (or clusters of activities). Each activity is associated with its *contract* that specifies (i) a conditions or a situation when this activity can be executed (ii) a situation that will result from this execution. As a result, instead of a flow of preordered tasks, DeCo specification describes:

- a set of tasks (with no explicit ordering) that must, should, or could be executed during the process;
- a set of rules allowing a dynamic task selection at a given process state from the list of tasks enabled at this state [7].

In a general case, the resulting process scenario is highly nondeterministic as each of the enabled task, if selected, will result in a different case development. Strictly speaking,

the concrete scenario of how a given case has been managed can be known only upon the process termination. We call it an execution trace.

4.2 The Role of Context

Dey in [35] defines a context as *".. any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*

As our example shows, mortgage approval process can vary strongly from country to country, agency to agency, and even from one mortgage application to another. These entities make a part of the process context and should be taken into consideration while configuring a concrete process scenario. More specifically, the context relevant to our process should encompass the characteristics of a mortgage lender and an applicant. The mortgage lender (a financial institution) can be characterized by its internal policies and adopted standards in customer services, risk management etc. For any concrete agency, also its resources available for process execution should be taken into consideration: number of employees, their roles, expertise, and responsibilities. Moreover, any financial institution should comply with some external regulations (e.g. federal law, etc.) that are defined by its country or state of residence. Thus, a geographical situation of a mortgage lender is also a context for mortgage approval process. The information about a mortgage applicant is explicitly handled by the process – it is a mortgage application file itself.

At design time, usually little is known about a process context: a CMP should comply with an industry standard, other external regulations and, if known, internal policies of an organization implementing it.

At deployment, the context is getting more explicit: agency type, its location, local resources and other specific facts about the process context allow designer to configure the process accordingly.

The final and the most specific part of the process context (or case context) is an application file – it appears and fills in during the mortgage approval execution. It allows configuring the process scenario in all details.

According to DeCo modeling approach, all the emerging context information should be constantly transformed into process configuration rules (at design, and deployment) or used to check the task compliance with such rules (at deployment and execution). Thus, the DeCo specification of mortgage approval process can be seen as a repository of tasks where each task can be instantiated for a given country, agency and even application case by answering the following questions:

-what resources (data, people, etc) are required for the task execution?

-what will be produced/modified upon termination of this task?

-what other constraints (e.g. functional, temporary, legacy, etc) must be fulfilled for the task execution?

DeCo does not provide means for modeling process context but rely strongly on the availability of context information. Coupling of DeCo with some context modeling approach will be explored in our future research.

4.3 Configurability at Multiple Perspectives

In the literature, several major perspectives of the process models are specified [8]: *the control flow perspective* that captures the temporal ordering of process tasks, events, and decision points; *the data perspective* that captures the lifecycle of data objects (creation, usage, modification, deletion) within the process; *the resource perspective* that describes how the process is carried out within the organization and deals with roles and resource assignments; *the operational perspective* that addresses the technical aspects of process execution and specifies the elementary process tasks and their assignment to concrete applications or application components of the organizations; *the context perspective* that describes the attributes related to the process execution context; *the performance perspective*, addressing the process cost effectiveness.

DeCo defines the concepts and semantics addressing process model configurability (i) on the *control flow* perspective by supporting *declarative style* and allowing non-deterministic execution scenarios; (ii) on the *data, operational,* and *resource* perspectives, providing the modeling notation for *configurable data objects, tasks, and roles* respectively; (iii) on the *context* perspective, providing *contextual configuration rules*.

Configurable roles in DeCo was already addressed in [36]. In this paper, we introduce the notation and semantics for configurable data objects and context-based configuration rules in DeCo. The other concepts will be addressed in detail in our future publications.

4.4 Configurable Data Objects and Their Semantics

A case of a foreign applicant recently arrived to the US following a new job assignment can become extremely difficult for a potential mortgage lender. The main problem with this case is that the applicant cannot provide the documents *required* by the standard process; instead, she is submitting *other* documents issued by a bank, an employer, or authority of her previous country of residence. Not matching the standards, such application can launch a long investigation process or can be simply rejected.

DeCo defines configuration mechanisms for data objects that include *optional* and *obligatory* data objects, specifies if a certain data artifact is *consumed*, *produced*, or *modified* by a given task and defines *synonym* and *alternative* relations between data objects as illustrated in Fig.1. These mechanisms help knowledge worker to anticipate the situation that she never met before, to process the data artifacts not “previewed” by a standard process scenario (if one exists) and to use them for more efficient decision-making.

Data object configuration in DeCo corresponds to configurability along the data perspective according to the taxonomy defined in [8].

Figure 1 illustrates the data object configuration diagram for *Tax forms* and *Tax return forms* required for formal mortgage application in the USA. A rectangle with thick outline and "**C**" in the right corner refers to a data object that can be further ***configured*** in the process model based on the situation or context.

By *alternatives* we specify the data objects that can replace the data object originally required by the task; *synonyms* are completely identical data objects used under different names by organizations or departments of one organization. A dashed

line with a tag "syn" depicts the synonym relation. A solid line with a tag "alt" depicts the alternative relation. More formally, these relations can be specified as follows:

Let $T(d_1,..,d_n)$ be a task where $d_1,..,d_n$ are the data objects consumed, produced, or modified by T.

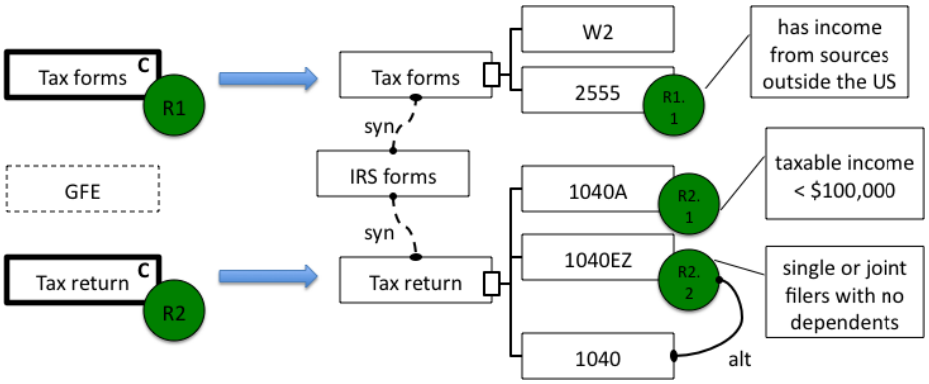

Fig. 1. Data object configuration in DeCo

Definition 1: Data object x is a *synonym* of d_i for a given process π iff it can replace d_i in T with no impact on π:

$$x \propto Syn_{\pi}(d_i) \Leftrightarrow T(..x,..) \Leftrightarrow T(..d_i..)$$

In Fig. 1 both tax forms and tax return forms can be called Internal Revenue Service (IRS) forms [source: wikipedia]. This is expressed using a synonym relation.

Definition 2: The fact that for a given process and task T the data object y is an *alternative* for d_i under a certain condition R implies that: If R holds then y can be accepted by T instead of d_i:

$$y \propto Alt_{\pi}(d_i,T,R) \propto R \rightarrow T(..y..)$$

In Fig.2, the form 1040EZ is a simplified form that can be applicable to single and joint filers with no dependents [source: wikipedia]. It can be considered as an alternative of the obligatory 1040 form. The rule 2.2 in this diagram explicitly specifies the condition where this alternative is applicable.

A data object d is *optional* for a given task T if it can be omitted in the contract of T. For example, the GFE certificate is optional for some countries and can be omitted in *Register Application* task. In DeCo, optional data objects are depicted with dashed borders.

Using the data object configuration, we can separate the management of data objects from the process scenario management.

Let us now come back to the foreign applicant example. Considering that the bank is willing to satisfy the foreign applicant's request, the main trouble for a loan officer would be to identify the relations between the standard forms provided by this applicant

but unknown for the current bank and the standard forms required by the mortgage approval process of this bank. Such investigation terminates with conclusions like the following:

- The form X provided by the applicant is an analogy of the form Y required by the process;
- The form X provided by the applicant can be accepted in place of the form Y required by the process under certain condition C;
- The form X provided by the applicant is irrelevant to the process.

This information can be expressed in terms of synonym, alternative relations or optional/obligatory property in DeCo:

- The form X provided by the applicant is an analogy of the form Y required by the process → *X is a synonym of Y within this process;*
- The form X provided by the applicant can be accepted in place of the form Y required by the process under certain condition C → *if condition C is met, then X is an alternative to Y within this process;*
- The form X provided by the applicant is irrelevant to the process → *X is optional within this process.*

The more synonym or alternative relations are determined, the more process model evolves, and the better decision-making support for the loan officer is provided.

4.5 Modeling Contextual Configuration Rules

One process often needs to be customized in order to meet the requirements of its deployment environment (e.g. country, state, corporate division) and/or to anticipate the concrete case circumstances [7]. Therefore, the possibility to enable/disable synonym and alternative relations and optional/obligatory properties of an element has to be provided not only at process design but also at customization and instantiation. For this purpose, DeCo defines context configuration rules (*context rules* for short).

In DeCo diagrams, context rules are depicted with dark circles labeled with a name of the rule. In the current version of DeCo, these rules are formulated as predicate expressions. If, based on the context, such predicate evaluates to *True,* then the corresponding configuration mechanism is enabled.

This concept can be illustrated on the following example: According to the banking regulations in the US, in certain cases the mortgage applicant may be asked to provide the lender with one or several supplementary tax forms and tax return forms. The diagram in Fig.1 illustrates the context rules implementing these regulations. Here the form W2 (Wage and Tax Statement) is an obligatory data object for the mortgage application; the form 2555 (Foreign Earned Income) should be provided by taxpayers who have earned income from sources outside the United States (a context rule 1.1). The form 1040A is limited to taxpayers with taxable income below $100,000 (this is expressed by the rule 2.1.). The form 1040EZ is a simplified form that can be applicable to single and joint filers with no dependents [source: wikipedia]. Under this condition, which can be expressed as a contextual rule 2.2, this form can be considered as an alternative of the obligatory 1040 form.

Context configuration rules can be defined based on (a) external regulations imposed by a concrete location (country, state, city) of the organization implementing the process, (b) internal policies specified at a company level (its country division, local branch or agency), (c) particular case conditions (e.g. a foreign applicant, sub-prime, first-time buyer, etc). Other context rules for the mortgage approval process may include: *isAvailable(InternalAppraisalAgent) = true; agencyLocation = North Carolina;* etc. The context rules should be specified for a process at design, refined at deployment and then controlled at execution.

5 Conclusion and Future Work

In this work the Declarative Configurable process specification language has been presented. The graphical notation of DeCo is based on BPMN, but similarities terminate here since, compared to BPMN, DeCo implements the declarative modeling principles. DeCo language is in its infancy. Validation of its modeling concepts and development of a modeling tool are the main milestones for our future research.

In this paper, we specified configurable data objects and contextual configuration rules. These concepts are integrated into DeCo process models. Other concepts defined in DeCo will be presented in our next publications.

Using the Mortgage approval process as an example, we illustrated how declarative modeling principles and configurability mechanisms can be used in order to improve the post-design process model adaptability - the characteristic utterly desired for knowledge-driven processes. Since the execution scenario of such processes cannot be predefined at design, *non-deterministic* declarative specifications become a natural solution.

Though encompassing a significant ambiguity and not suitable for simulation, declarative process specifications may serve a useful tool for process validation and verification – the techniques known from software engineering [31]. As soon as new context information emerges, the declarative process specification evolves and ideally becomes more and more deterministic. This evolution can be compared to a *step-wise refinement* of software specifications [33]; the notion of refinement for graphical specifications is presented in [30]. The step-wise process refinement and its validation in DeCo will be addressed in our future publications.

Context rules play an important role in process configuration. Providing that an organization can be exposed to various sources of regulations, thousand context rules will emerge in a process model. This increases a risk of conflicting rules leading to the process model inconsistency. DeCo provides a FOL-based semantics for the context rules. The next step of our research will be focused on modeling, validation and verification of context rules.

The modeling method presented in this work is in its infancy. The modeling notation still requires major improvement in order to be adopted by practitioners. However the most important issue for us is scalability and validity of the DeCo process specifications. Modeling more elaborated (real life) case management process with DeCo and validation of results is a critical milestone in our research.

References

1. Smith, H., Fingar, P.: Business Process Management: The Third Wave. Meghan-Kiffer Press (2003)
2. Barjis, J.: The Importance of Business Process Modeling in Software Systems Design. Journal of the Science of Computer Programming 71(1), 73–875 (2008)
3. Hill, J.B., Lheureux, B.J., Olding, E., Plummer, D.C., Rosser, B., Sinur, J.: Predicts 2010: Business Process Management Will Expand Beyond Traditional Boundaries, http://www.gartner.com/resId=1231219
4. OMG, Case Management Process Modeling (CMPM) Request For Proposal: Bmi/2009-09-23
5. Swenson, K.D.: Mastering the Unpredictable. How adaptive case management will revolutionize the way the knowledge workers get things done. Meghan-Kiffer Press (2010)
6. de Man, H.: Case Management: A Review of Modeling Approaches, BPTrends (January 2009)
7. Rychkova, I., Nurcan, S.: Towards Adaptability and Control for Knowledge-Intensive Business Processes: Declarative Configurable Process Specifications. In: Proc. 44th Annual Hawaii International Conference on System Sciences, HICSS, pp. 1–10. IEEE (2011)
8. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press (1996)
9. Barjis, J., Rychkova, I., Yilmaz, L.: Modeling and Simulation Driven Software Development. In: Chinni, M.J., Weed, D. (eds.) Spring Simulation Multi Conference, pp. 4–10 (2011)
10. OASIS Web Services Business Process Execution Language, WSBPEL (2006)
11. van der Aalst, W.M.P.: Pi calculus versus Petri nets: Let us eat "humble pie" rather than further inflate the "Pi hype". BP Trends 3(5), 1–11 (2005)
12. van der Aalst, W.M.P.: Making Work Flow: On the Application of Petri Nets to Business Process Management. In: Esparza, J., Lakos, C. (eds.) ICATPN 2002. LNCS, vol. 2360, pp. 1–22. Springer, Heidelberg (2002)
13. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information and Software Technology 50, 1281–1294 (2008)
14. Barjis, J.: Automatic Business Process Analysis and Simulation Based on DEMO. J. Enterprise Information Systems 1(4), 365–381 (2007)
15. Dietz, J.L.G.: Enterprise Ontology –Theory and Methodology. Springer, New York (2006)
16. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science – Research and Development 23(2), 99–113 (2009)
17. Yu, E.S.K., Mylopoulos, J.: Understanding "why" in software process modeling, analysis, and design. In: The Proceedings of ICSE 1994, pp. 159–168 (1994)
18. Fahland, D., Lübke, D., Mendling, J., Reijers, H., 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.) BPMDS 2009 and EMMSAD 2009. LNBIP, vol. 29, pp. 353–366. Springer, Heidelberg (2009)
19. Wegmann, A.: On the Systemic Enterprise Architecture Methodology (SEAM). In: Proc. 5th International Conference on Enterprise Information Systems, pp. 483–490 (2003)

20. Wegmann, A., Lê, L.-S., Regev, G., Woods, B.: Enterprise Modeling Using the Foundation Concepts of the RM-ODP ISO/ITU. Standard Information Systems and E-Business Management, vol. 5, pp. 397–413 (2007)
21. Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modeling. In: Requirements Engineering, vol. 4(4). Springer, London (1999)
22. Nurcan, S., Etien, A., Kaabi, A., Zoukar, I., Rolland, C.: A Strategy Driven Business Process Modelling Approach. Special issue of the Business Process Management Journal on Goal-Oriented Business Process Modeling (2005)
23. Korherr, B., List, B.: Extending the EPC and the BPMN with Business Process Goals and Performance Measures. In: ICEIS, vol. (3), pp. 287–294 (2007)
24. Awad, A.: BPMN-Q: A Language to Query Business Processes. In: EMISA, pp. 115–128 (2007)
25. La Rosa, M., Dumas, M., ter Hofstede, A.H.M., Mendling, J.: Configurable multi-perspective business process models. Journal Information Systems 36(2) (2011)
26. La Rosa, M., Dumas, M., ter Hofstede, A.H.M., Mendling, J., Gottschalk, F.: Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 199–215. Springer, Heidelberg (2008)
27. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
28. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: Full Support for Loosely-Structured Processes. In: Spies, M., Blake, M.B. (eds.) Proc.11th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 287–298. IEEE (2007)
29. Rychkova, I., Regev, G., Wegmann, A.: Using Declarative Specification. In: Business Process Design. International Journal of Computer Science & Applications (2008)
30. Rychkova, I.: Formal semantics for refinement verification of entreprise models. Dir.: Alain Wegmann. Thèse EPFL, no 4210 (2008), http://library.epfl.ch/theses/?nr=4210
31. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press (2006)
32. Rychkova, I., Nurcan, S.: The Old Therapy for the New Problem: Declarative Configurable Process Specifications for the Adaptive Case Management Support. In: zur Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 420–432. Springer, Heidelberg (2011)
33. Wirth, N.: Program development by stepwise refinement. Communications of the ACM (1971)
34. BPMI/OMG, Inc. Business Process Modeling Notation. Version 1.0 (February 6, 2006), http://www.bpmn.org/
35. Dey, A.: Understanding and Using Context. Personal and Ubiquitous Computing 5, 4–7 (2001)
36. Denekere, R., Rychkova, I., Nurcan, S.: Modeling the role variability in the MAP process model. In: Proc. RCIS (2011)
37. Rolland, C., Nurcan, S.: Business Process Lines to deal with the Variability. In: Hawaii International Conference on System Sciences (HICSS), Hawaii, USA (January 2010)