

Comparing Declarative Process Modelling Languages from the Organisational Perspective

Stefan Schönig^(✉) and Stefan Jablonski

University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany
{stefan.schoenig,stefan.jablonski}@uni-bayreuth.de

Abstract. The spectrum of business processes can be divided into two types: well-structured routine processes and agile processes with control flow that evolves at run time. In a similar way, two different representational paradigms can be distinguished: procedural models and declarative models which define rules that a process has to satisfy. Agile processes can often be captured more easily using a declarative approach. While in procedural languages the organisational perspective can be modelled adequately, in declarative languages, however, an adequate representation of organisational patterns is often still not possible. Agile processes, however, need to explicitly integrate organisational coherencies due to the importance of human decision-making. This paper presents a review of declarative modeling languages, outlines missing aspects and suggests research roadmaps for the future.

Keywords: Declarative process modelling · Organisational perspective · Process modelling languages

1 Introduction

A well accepted method for structuring the activities carried out in an organisation is business process management (BPM). BPM usually involves modelling, executing and analysing business processes [1]. The spectrum of business processes can be divided into two types [2]: well-structured routine processes with exactly prescribed control flow and agile processes with control flow that evolves at run time without being exactly predefined a priori. Agile processes are common in healthcare where, e.g., patient diagnosis and treatment processes require flexibility to cope with unanticipated circumstances. In a similar way, two different representational paradigms can be distinguished: procedural models describe which activities can be executed next and declarative models define execution constraints that the process has to satisfy. The more constraints are added to the model, the less possible execution alternatives remain. As agile processes may not be completely known a priori, they can often be captured more easily using a declarative rather than a procedural modelling approach [3–5].

Independent from a specific modeling paradigm different perspectives on a process exist [6]. The organisational perspective, often also referred to as resource

perspective deals with the definition and the allocation of human and non-human resources to activities [1]. In procedural process modelling languages like BPMN resource assignments can be expressed by extension languages with well-defined semantics like, e.g., the Resource Assignment Language (RAL) [7]. Furthermore, implementations like the YAWL system [8] provide rich support for the resource perspective, too. Lately, with RALph [9] even a graphical notation for defining human resource assignments in procedural process models with semantics defined in RAL has been introduced.

Due to the importance of human decision-making and expert knowledge, especially agile processes need to explicitly integrate the organisational perspective [5, 10]. Unlike for procedural languages, however, the support of resource management in current declarative process modelling languages is ambiguous. While several studies of resource assignment approaches in procedural modelling languages exist [7, 9], an in-depth analysis of the representation of organisational aspects in declarative process modelling languages is still missing. In this paper, we fill this gap by studying capabilities and expressiveness of existing executable declarative modelling languages and frameworks like Declare [3], the Declarative Process Intermediate Language (DPIL) [11] or the new declarative modelling standard Case Management Modelling and Notation (CMMN) [12] according to the well-known Workflow Resource Patterns (WRPs) [13]. Therefore, the contribution of this paper is a survey and comparison of modelling capabilities and insufficiencies of existing declarative process modelling languages. As an outlook we also suggest a solution approach to overcome existing issues.

The remainder of the paper is structured as follows: Sect. 2 describes the criteria, the evaluation framework and a running example. In Sect. 3, we give an in-depth analysis of the representation of the organisational perspective in well-known declarative process modelling languages. The results and possible solutions to the identified gaps are discussed in Sect. 4. Finally, Sect. 5 concludes this work.

2 Evaluation Framework and Running Example

In this section, we specify the evaluation framework that we will use for the analysis. Furthermore, we present a simplified example process to illustrate issues.

2.1 Evaluation Criteria

The expressiveness of modelling languages is assessed according to the well-known workflow resource patterns [13]. Specifically, we use the creation patterns, as they are related to resource selection. The patterns include:

- *Direct Allocation* is the ability to specify at design time the identity of the resource that will execute a task.
- *Role-Based Allocation* is the ability to specify at design time that a task can only be executed by resources that correspond to a given role.

- *Organisational Allocation* is the ability to offer or allocate activity instances to resources based their organisational position and their relationship with other resources.
- *Separation of duties* is the ability to specify that two tasks must be allocated to different resources in a given process instance.
- *Case Handling* is the ability to allocate the activity instances within a given process instance to the same resource.
- *Retain Familiar* is the ability to allocate an activity instance within a given process instance to the same resource that performed a preceding activity instance, when several resources are available to perform it. This pattern is also known as *binding of duties*.
- *Capability-Based Allocation* is the ability to offer or allocate instances of an activity to resources based on their specific capabilities.
- *Deferred Allocation* is the ability to defer specifying the identity of the resource that will execute a task until run time.
- *History-Based Allocation* is the ability to offer or allocate activity instances to resources based on their execution history.

Note that creation patterns Authorisation and Automatic Execution are not on the list. The former is excluded since it is not related to the definition of conditions for resource selection, and the latter since it is not related to the assignment language and is inherently supported by all Business Process Management Systems (BPMSs). Furthermore, we consider the direct allocation possible if a role-based allocation is supported.

In agile processes human participants are the main drivers of process execution [14]. For actors in different organisational groups frequently different rules apply. This results in different temporal dependencies between activities for different persons. In addition to resource assignment we also have to focus on patterns that relate to the control flow and the organisational perspective at the same time. These coherencies are called *cross-perspective* [15]. Cross-perspective patterns can be found in different application areas, e.g., in cases where the execution of certain process steps is bind to conditions only for certain actors. Examples are: (i) while in general work results like, e.g., documents, do not need to be checked before publication, the publication in case of trainees requires the work results to be checked in advance and (ii) while students in higher semesters can directly start lectures, new students need to complete a consulting dialogue before.

2.2 Running Example

Throughout this paper, we will use the simplified business trip process in a university as a running example. Some of the organisational patterns described above can be easily demonstrated by means of this simple example. For carrying out a business trip the applicant needs to fill out an application thats needs to be approved afterwards. Furthermore, at some point a flight needs to booked. Here, several organisational patterns can be identified:

- *Role-Based Allocation*: in any case the application needs to be checked and approved by an employee of the administration department.
- *Retain Familiar*: the applicant himself knows circumstances and appointments best. Therefore, the flight must be booked by the applicant himself.
- *Cross-Perspective Pattern, Role-Based Sequence*: professors can book flights at any time and do not need to wait for the approval. Students, however, have to stick to a certain execution order of activities, i.e., they need to wait for the approval before a flight can be booked. Here, the control-flow is directly influenced by organisational circumstances.

3 Analysis

In this section, we analyse the capabilities of declarative process modelling languages w.r.t. the organisational patterns described in Sect. 2, i.e., resource assignment patterns as well as cross-perspective patterns. Our analysis comprises a selection of five well-known declarative process modeling languages, i.e., *Declare* [16], *DCR-Graphs* [17], *CMMN* [12], *DPIL* [11] and *EM-BrA²CE* [18]. To the best of the authors knowledge this set contains the most recent *executable* and system supported rule-based process modelling languages. Recently also hybrid modelling approaches that combine procedural and declarative elements like, e.g., BPMN-D [19] or a combination of petri net and Declare modelling elements [20] have been proposed. Since these approaches only focus on control flow, organisational patterns are not yet part of the language specifications. Therefore, these languages do not need to be considered in our analysis.

3.1 Declare

Declare is a framework for modelling and executing rule-based process models. The framework allows for the definition of rule templates and languages. The most frequently used Declare languages are *ConDec* [16] and *DecSerFlow* [21]. In order to execute a model it has to be transformed to an interpretable formalism. Within Declare two formalisms have been used so far: *linear temporal logic (LTL)* and the *event calculus*. Therefore, for each rule template the formalized expression needs to be given. This way, process models can be translated into a validate and executable logical form. An example is the rule “in order to perform b , a must have been performed before” that is frequently captured by the macro *precedence*(a , b). Additionally, Declare provides a graphical representation of rules. The *precedence* macro, e.g., is visualized by an arrow with a point. The transformation of the *precedence* macro to a LTL expression is given by $(\neg B)WA$. The symbol W , e.g., states that $\neg B$ has to hold until A occurs. For execution the LTL formula is transformed to an automaton [16]. The organisational perspective, however, is only rudimentary implemented in Declare. Independently of specific models the Declare system allows for the definition of simple organisational models, i.e., users and roles [16]. A definition of individual relations between organisational elements is not possible.

During the specification of concrete models the user can assign one or more roles to activities. At runtime, activities are then offered to users in the assigned roles for execution. Hence, Declare only allows for the definition of a direct or a role-based task allocation. More complex organisational patterns like, e.g., binding or separation of duties cannot be modelled and executed by Declare. Declare only constrains the starts of activities and interrelates them temporally. However, a rule as a whole may depend on arbitrary conditions including the data context. The rule *precedence*(a, b) with the condition $x < 3$ claims that a must have been performed before b can be started only if $x < 3$. However, a coherency like “before publication the document of a student must be reviewed by his supervisor” cannot be expressed in this way. Hence, Declare does not support cross-perspective modelling.

Figure 1 visualizes the example process in Declare. The control flow can easily be captured by a *precedence* rule between the activities “apply for trip” and “approve application”. However, roles and resources are not part of the graphical notation but are defined in the Declare system. This way, “apply for trip” and “book flight” can be carried out by all the defined resources while “approve application” can only be performed by persons with role “administration”. The binding of duties between application and booking cannot be modelled in Declare. Furthermore, it is not possible to combine the control flow and organisational circumstances, i.e., the fact that only students need to stick to a certain execution order cannot be modelled.

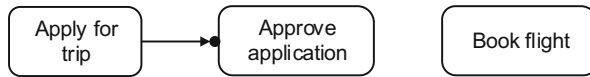


Fig. 1. Agile example process modelled in Declare

3.2 DCR-Graphs

Dynamic Condition Response-Graphs (DCR-Graphs) [17] is a declarative process modelling language similar to Declare. The graphical representation of events and rules relates to Declare. However, the number of rule templates in DCR graphs is considerably smaller than in Declare. Activities can only be connected by rules of the types *Condition*, *Response*, *Milestone*, *Inclusion* and *Exclusion*. Nevertheless, DCR graph models reach the same expressiveness as Declare. DCR graph models are not relying on a LTL formalism but are directly interpreted and executed based on a transformation to Buechi automata. Like in Declare the organisational perspective in DCR graphs is limited to the definition and assignment of resources and roles [17]. Modeling of complex allocation rules is currently not possible. Since only the rule types mentioned above can be modelled between activities, cross-perspective relations are not supported, too.

Therefore, the influence of organisational coherencies on the control flow cannot be modelled in DCR graphs.

Figure 2 shows the example process in DCR graph notation. The temporal dependency between “apply for trip” and “approve application” can be modelled by a *Condition* rule that relates to a *precedence* rule in Declare. In contrast to Declare, role-based allocations are part of the graphical notation. Hence, the model shows that applications need to be approved by resources with role “administration”. The binding of duties between the application and the booking as well as the cross-perspective temporal dependency between the approval and the booking cannot be modelled with DCR graphs.

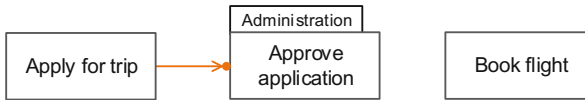


Fig. 2. Agile example process modelled as DCR graph

3.3 Case Management Modeling and Notation (CMMN)

The *Case Management Model and Notation (CMMN)* [12] has been introduced by the Object Management Group (OMG) in 2013 and is based on the *Guard-Stage-Milestone (GSM)* model [22]. In CMMN a case is considered as an agile process. A case is structured into stages and contains Tasks, i.e., activities and CaseFileItems, i.e., data objects and documents. Rules are defined through so-called Sentries that form a triplet comprising an event, a condition and an action. If the specified event occurs and a certain condition is satisfied then the defined action is triggered. Here, events can refer to transitions of data objects as well as Tasks and Milestones. The condition of a Sentry, however, can only refer to data objects. CMMN neglects the organisational perspective. The potential performer of a human task can only be selected on the basis of a role and the perspective is completely missing in the graphical representation of CMMN models (diagrams). Therefore, only a direct and a role-based allocation pattern can be modelled in CMMN. Cross-perspective modelling is limited in CMMN, too. Rules may only depend on events of informational entities, i.e., case file items and activities and may only constrain the entry and exit of activities. They may not, e.g., depend on or constrain organisational aspects.

Figure 3 shows the example process in CMMN. The temporal dependency between “apply for trip” and “approve application” is modelled by a dashed arrow with a diamond (Sentry). Since CMMN only allows for the definition of roles. Hence, it can be defined that the task “approve application” must be performed by a person with role “administration”. However, this is not visualized in the graphical notation. Furthermore, it is not possible to model the

binding of duties. Since Sentries can only refer to Tasks and CaseFileItems, cross-perspective coherencies like the one between “apply for trip” and “book flight” cannot be modelled in CMMN.

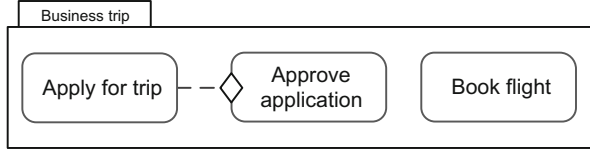


Fig. 3. Agile example process modelled in CMMN

3.4 Declarative Process Intermediate Language (DPIL)

The textual *Declarative Process Intermediate Language (DPIL)* [11] is a multi-perspective rule-based modeling language. It allows representing several business process perspectives, namely, control flow, data and resources. The expressiveness of DPIL and its suitability for business process modelling have been evaluated [11] with respect to the Workflow Resource Patterns [13]. Some of the elements of a DPIL process model undergo a life cycle composed of events that is managed by the execution engine. A human task, e.g., can be started and completed while a data object can be read or written. Besides the static elements like human tasks and data objects, a process model may specify rules constraining that series of events. It may, e.g., claim that some data object may only be written after some task has been started. DPIL provides a textual notation based on the use of *macros* to define reusable rules. For instance, the *sequence* macro (*sequence(a,b)*) states that the existence of a *start event* of task *b* implies the previous occurrence of a *complete event* of task *a*.

In order to express organisational relations, DPIL builds upon a generic organisational meta model that has been described in [23] and is depicted in Fig. 4a. It comprises the following elements: *Identity* represents agents that can be directly assigned to activities, i.e., both human and non-human resources. *Group* represents abstract agents that may describe several identities as a whole, e.g., roles or groups. *Relation* represents the different relations (*RelationType*) that may exist between these elements. It is suitable to define, e.g., that an identity has a specific role, that a person is the boss of another person, or that a person belongs to a certain department. Figure 4b illustrates an exemplary organisational model of a university research group. It is composed of two roles (Professor, Student) assigned to three people (SJ, SS, BR) and several relations between them indicating who is supervised by whom. Based on the described organisational meta model, DPIL allows for modelling basic, i.e., direct and role-based allocation as well as more complex organisational patterns, i.e., separation and binding of duties as well as capability-based and organisational allocation patterns. Since in DPIL it is not possible to specify relations that relate

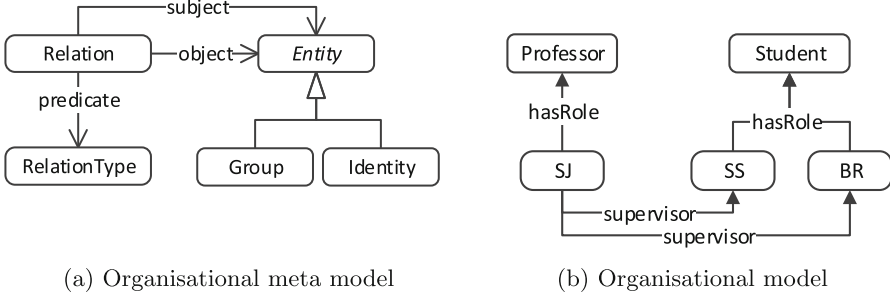


Fig. 4. Organisational meta model and example organisational model

to other process instances, the history-based allocation pattern cannot be modelled. Rules in DPIL can relate different perspectives at the same time. Hence, cross-perspective modelling is unrestricted possible.

```

use group Administration
use group Student
use relationtype hasRole

sequence(a,b) iff start(of b at :t) implies complete(of a < t)
binding(a,b) iff start(of a by :p) and start(of b) implies start(of b by p)
role(t,r) iff start(of t by :p) implies
    relation(subject p predicate hasRole object r)
roleSequence(a,b,r) iff start(of b by :i at :t) and
    relation(subject i predicate hasRole object r)
    implies complete(of a at < t)

process Business Trip {
    task Apply for trip
    task Approve application
    task Book flight

    ensure sequence(Apply for trip, Approve application)
    ensure role(Approve application, Administration)
    ensure binding(Apply for trip, Book flight)
    ensure roleSequence(Approve application, Book flight, Student)
}

```

Fig. 5. Process for trip management modelled with DPIL

Figure 5 shows the example process modelled in DPIL. The first three lines refer to roles and relation types of the organisational model. In the following lines four different macros are defined, i.e., the macros *sequence*, *binding*, *role* and *roleSequence*. The *sequence* macro relates to the *precedence* template of Declare. The *role* macro defines a role-based task allocation. The *binding* macro

defines the structure of a binding of duties pattern. The *roleSequence* finally depicts a cross-perspective rule template, i.e., the start of activity *b* by an actor in role *r* requires the completion of activity *a* before. The actual process model starts with the definition of the three tasks. The temporal dependency between application and approval is modelled by use of the *sequence* macro. The role-based task sequence is modelled by instantiating the *role* macro. Furthermore, the binding of duties between trip application and flight booking can be defined by the usage of the *binding* macro. The fact that only students have to stick to a certain execution order is modelled by using the *roleSequence* macro, i.e., booking a flight by a student implies the approval of the application before.

3.5 EM-BrA²CE

The EM-BrA²CE framework [18,24] supports declarative process modeling by relying on a vocabulary based on the OMG Semantics for Business Vocabulary and Business Rules (SBVR) specification. As the EM-BrA²CE vocabulary extends the conceptual vocabularies of the SBVR, it supports the specification of highly expressive statements, concerning business concepts and rules of the organisation, the involved interacting agents and the events characterizing the execution of the systems. The SBVR specification defines a structured, english vocabulary for describing and verbalizing rules. For execution, the SBVR rules are translated to event-condition-action (ECA) rules using templates. Therefore, processes are effectively modelled using the ECA templates. The organisational perspective in EM-BrA²CE implements the existing standard for Role-Based Access Control (RBAC) patterns, i.e., it is possible to model separation and binding of duties as well as the case handling pattern. Since SBVR is used as an ontology language, agents can be described with arbitrary relations and attributes that rules can directly refer to [24]. Therefore, also capability-based and organisational allocation patterns can be expressed. Like in DPIL, modeling of history-based patterns is not possible in EM-BrA²CE. The following SBVR rules depict the example process in EM-BrA²CE notation:

An approve application activity1 can only start after a apply for trip activity2 has completed.

An agent that *has role* administration can perform approve application activity. It is necessary that an agent that *has been assigned* to an apply for trip activity performs a book flight activity.

An agent that *has role* student can only perform an book flight activity1 after a approve application activity2 has completed.

4 Discussion and Solution Approach

The results of the analysis are summarized in Table 1. The table highlights a considerable discrepancy between the expressiveness of graphical and textual

Table 1. Organisational patterns in declarative modelling languages

organisational Pattern	Declare	DCR	CMMN	DPIL	EM-BrA ² CE
Direct Allocation	✓	✓	✓	✓	✓
Role-based Allocation	✓	✓	✓	✓	✓
organisational Allocation	-	-	-	✓	✓
Separation of Duties	-	-	-	✓	✓
Case Handling	-	-	-	✓	✓
Binding of Duties	-	-	-	✓	✓
Capability-Based Allocation	-	-	-	✓	✓
Deferred Allocation	-	-	-	✓	✓
History-Based Allocation	-	-	-	-	-
Organisation-Based Control Flow (Cross-Perspective Rules)	-	-	-	✓	✓
Graphical Notation	✓	✓	✓	-	-

declarative process modelling languages. While the biggest part of organisational patterns cannot be expressed in current graphical notations, the analysed textual languages offer rich support for modelling both complex resource assignment patterns and cross-perspective coherencies. In DPIL and EM-BrA²CE, only the instance spanning history-based allocation pattern cannot be expressed. However, the textual notations are generally more difficult to model and to understand. It is widely accepted that visual notations can be beneficial for system development [25]. Hence, a visual notation with well-defined semantics for declaratively modelling organisational aspects in an integrated way is convenient.

It is striking that the development of a new graphical notation is not essential. In order to bridge the gap, we propose to map an existing and well-evaluated visual notation to one of the analysed textual process modelling notations. Here, we consider the extended Compliance Rule Graphs (eCRG) [26] a appropriate candidate notation. eCRG is a visual language for modeling compliance rules that not only covers the control flow perspective, but provides integrated support for the resource perspective as well. Note, that eCRG is not an executable declarative process modelling language but represents process compliance rules. That is why the language is not considered in our analysis. In eCRG, however, it is possible to graphically model cross-perspective patterns.

Figure 6 shows the different organisational patterns of the example process depicted as a eCRG model. The binding of duties between trip application and flight booking as well as the role-based allocation for the approval task can be adequately modelled with eCRG and are visualized in Fig. 6a and b, respectively. The role-based task sequence is modelled as a eCRG model in Fig. 6c. The eCRG model states that whenever *Book flight* is performed by a *Student* then *Approve application* must have been performed before. The concrete mapping of eCRG to, e.g., DPIL should hence be up to future research.

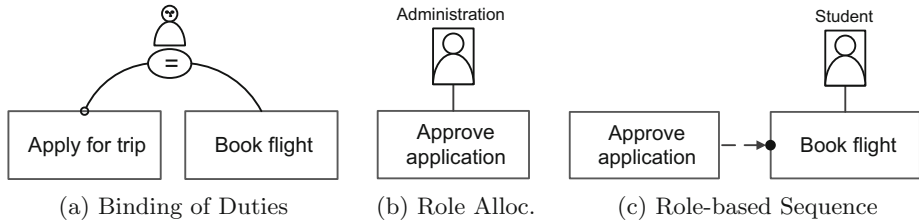


Fig. 6. Organisational patterns of the example process in eCRG

5 Conclusion

The work at hand presents an in-depth analysis of the support for the organisational perspective in existing declarative process modeling languages. We studied and outlined the capabilities and expressiveness of several well-known rule-based modeling languages and frameworks according to well-defined criteria and identified missing aspects. The results of our analysis show a considerable discrepancy between the capabilities of graphical and textual notations. While the biggest part of organisational patterns cannot be expressed in current graphical notations, the analysed textual languages offer rich support for modelling both complex resource assignment patterns and cross-perspective coherencies. With a language like, e.g., DPIL almost every organisational coherency can be modelled, however, it does not offer a graphical notation which is essential for a language to be adequately usable for end users. In order to close the gap, we propose to map an existing and well-evaluated visual notation to one of the analysed textual process modelling notations. Here, we consider the extended Compliance Rule Graphs (eCRG) notation as an adequate candidate. The concrete mapping of eCRG to, e.g., DPIL should hence be up to future research.

References

1. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2013)
2. Jablonski, S.: MOBILE: A modular workflow model and architecture. In: Working Conference on Dynamic Modelling and Information Systems (1994)
3. van der Aalst, W., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. *Comput. Sci. Res. Dev.* **23**(2), 99–113 (2009)
4. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.: Imperative versus declarative process modeling languages: An empirical investigation. *Business Process Management Workshops*, pp. 383–394 (2012)
5. Vaculín, R., Hull, R., Heath, T., Cochran, C., Nigam, A., Sukaviriya, P.: Declarative business artifact centric modeling of decision and knowledge intensive business processes. In: Enterprise Distributed Object Computing Conference (EDOC), no. Edoc, pp. 151–160 (2011)
6. Jablonski, S., Bussler, C.: Workflow management: modeling concepts. architecture and implementation (1996)

7. Cabanillas, C., Resinas, M., del Río-Ortega, A., Ruiz-Cortés, A.: Specification and automated design-time analysis of the business process human resource perspective. *Inf. Syst.* **52**, 55–82 (2015)
8. Van der Aalst, W.M., Ter Hofstede, A.H.: Yawl: yet another workflow language. *Inf. Syst.* **30**(4), 245–275 (2005)
9. Cabanillas, C., Knuplesch, D., Resinas, M., Reichert, M., Mendling, J., Ruiz-Cortés, A.: RALph: a graphical notation for resource assignments in business processes. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) *CAiSE 2015. LNCS*, vol. 9097, pp. 53–68. Springer, Heidelberg (2015)
10. Van der Aalst, W.M., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. *Data Knowl. Eng.* **53**(2), 129–162 (2005)
11. Zeising, M., Schöning, S., Jablonski, S.: Towards a common platform for the support of routine and agile business processes. In: *Collaborative Computing: Networking, Applications and Worksharing* (2014)
12. Object Management Group (OMG). *Case Management Model and Notation (CMMN)*, Version 1.0 (2014)
13. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: identification, representation and tool support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) *CAiSE 2005. LNCS*, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
14. Vaculin, R., Hull, R., Vukovic, M., Heath, T., Mills, N., Sun, Y.: Supporting collaborative decision processes. In: *International Conference on Services Computing*, pp. 651–658 (2013)
15. Schöning, S., Cabanillas, C., Jablonski, S., Mendling, J.: Mining the organisational perspective in agile business processes. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) *BPMDs 2015 and EMMSAD 2015. LNBIP*, vol. 214, pp. 37–52. Springer, Heidelberg (2015)
16. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) *BPM Workshops 2006. LNCS*, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
17. Hildebrandt, T., Mukkamala, R.R., Slaats, T., Zanitti, F.: Contracts for cross-organizational workflows as timed dynamic condition response graphs. *J. Logic Algebraic Program.* **82**(5), 164–185 (2013)
18. Goedertier, S., Vanthienen, J.: Declarative process modeling with business vocabulary and business rules. In: Meersman, R., Tari, Z. (eds.) *OTM-WS 2007, Part I. LNCS*, vol. 4805, pp. 603–612. Springer, Heidelberg (2007)
19. De Giacomo, G., Dumas, M., Maggi, F.M., Montali, M.: Declarative process modeling in bpmn, In press (2015)
20. Westergaard, M., Slaats, T.: Mixing paradigms for more comprehensible models. In: Daniel, F., Wang, J., Weber, B. (eds.) *BPM 2013. LNCS*, vol. 8094, pp. 283–290. Springer, Heidelberg (2013)
21. 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)
22. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., Vaculin, R.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Bravetti, M. (ed.) *WS-FM 2010. LNCS*, vol. 6551, pp. 1–24. Springer, Heidelberg (2011)
23. Bussler, C.: *Organisationsverwaltung in Workflow-Management-Systemen*. Dt. Univ.-Verlag (1998)

24. Goedertier, S.: Declarative techniques for modeling and mining business processes. Ph.D. thesis, Katholieke Universiteit Leuven (2008)
25. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. *Softw. IEEE* **31**(3), 79–85 (2014)
26. Semmelrodt, F., Knuplesch, D., Reichert, M.: Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In: Bider, I., Gaaloul, K., Krogstie, J., Nurcan, S., Proper, H.A., Schmidt, R., Soffer, P. (eds.) *BPMDs 2014 and EMMSAD 2014*. LNBIP, vol. 175, pp. 48–63. Springer, Heidelberg (2014)

Business Process Management Workshops
BPM 2015, 13th International Workshops, Innsbruck,
Austria, August 31 – September 3, 2015, Revised
Papers
Reichert, M.; Reijers, H.A. (Eds.)
2016, XIII, 596 p. 203 illus., Softcover
ISBN: 978-3-319-42886-4