REFlex: an entire solution to business process modeling

Renata M. de Carvalho and Natália C. Silva

[1] University of Quebec at Montreal, LATECE Laboratory, Canada,
renatawm@gmail.com
[2] C.E.S.A.R - Recife Center for Advanced Studies and Systems, Recife, Brazil
naatcabral@gmail.com

Abstract. Some approaches have emerged providing flexibility to business processes (BP) execution. Among them, we can enhance the declarative one, which purpose is to model *what* must be done without describing *how* they must execute. Previously, i) we extended the definition of the business process to approach web service orchestration, besides the definition of activities and rules; and ii) we formalized the REFlex rule engine, allowing us to discuss the verification of the engine core properties and the *liveness-enforcing* mechanism properties. In this paper, we aim at discussing and presenting the REFlex as an entire solution to BPM. We show that, using REFlex, an user has support to: i) model a business process; ii) trust in a formalized semantics that does not allow the user to be guided to an unacceptable situation; and iii) link (some) activities to web services, (semi-)automating the process execution.

Key words: Declarative business process, rule engine, service orchestration, formalization, flexibility

1 Introduction

The purposes for companies to model their business processes (BPs) have been presented in different works [25, 18]. Languages such as BPMN (Business Process Modeling Language [26]) have been used to provide a graphical representation of the BP and its concepts. But such kind of representation defines not only a set of actions, but also in which order and by whom each action must be performed.

Some approaches have emerged to provide flexibility to BP execution [15]. Among them, we can enhance the declarative one. The purpose of a declarative BP is to model *what* must be done (the actions) without describing *how* they must execute [17]. This is possible by defining a set of activities (actions) in a company and a set of business rules (constraints). Any activity is allowed to execute since its execution does not violate any of the defined constraints. In that way, the order of the process execution is defined in runtime.

The declarative processes are useful for companies whose business logic is constantly changing. Moreover, some authors [25] believe that reading and understanding a business rule is more natural for employees with few technical IT

skills. Therefore, a declarative process may be used to communicate between different layers of a company.

In a previous work [2], we proposed REFlex, a graph-based rule engine to execute declarative processes. In other work [20] we extended the definition of the business process to approach web service orchestration, besides the definition of activities and rules. We described how REFlex represents a declarative BP execution as a graph. The graph is the representation only of the current execution step, and it is updated as the activities are executed. The graph itself and some annotations added allow its compliance with the described rules. We also showed that storing all information necessary to control the process execution in activities (nodes) and rules (edges) allows for an efficient implementation of a declarative BP engine [3]. In terms of space complexity, REFlex avoids the well known problem of state space explosion.

At a given execution point, the state of a declarative process is defined by the state of its activities. An activity can assume three different states: *enabled* - can execute; *disabled* - cannot execute at this point; *blocked* - cannot execute anymore. After the execution of an activity, the rule engine evaluates the current process state and the business rules to determine the next state of the process. The engine also analyzes global properties of the process. For instance, if the business rules require the execution of a set of activities, the engine allows the process to terminate only after executing these activities.

Other solutions to execute declarative processes exists, such as Declare and DCR Graphs. Each one has its own properties and a different manner to represent and execute a process. But none of them support the integration of the business process to web services. Hence, this paper aims at discussing the REFlex approach as an entire solution to business process modeling. Different from the others, using REFlex, an user has support to: i) model a business process; ii) trust in a formalized semantics that does not allow the user to be guided to an unacceptable situation; and iii) link (some) activities to web services, (semi-)automating the process execution.

This paper is structured as follows: Section 2 provides an overview of the background for this work; Section 3 discuss the REFlex solution, and shows how REFlex is inserted in the BPM area; Section 4 presents an overview of works related to our research; and final remarks and a conclusion appear in Section 5.

2 Flexible Processes

When company tasks are less repetitive and predictable, an approach that determines the order of tasks execution may not be the best adopted practice [15]. Using such an approach, a business logic could be too simple and unable to handle all situations; or too complex and hard to maintain because it tries to model all possible situations. Both situations present problems to the company. Flexible processes emerged to deal with these problems. To achieve the desired flexibility, the business logic must let the stakeholders to decide and perform

activities in any order, since the sequence (or part of that) was not explicitly prohibited.

In this context we can enhance the declarative approach, which is the flexible approach used in this work. The purpose of a declarative process is to define the BP in terms of business rules using a declarative language. While in other approaches we must specify all execution alternatives, a declarative model is guided by constraints [16]. Any activity that does not violate a constraint can be executed. Hence, adding new constraints reduces the number of execution options.

In a declarative BP, two elements compose a model: activities and business rules. An activity represents an action taken to modify the state of the modeled system. An activity must be performed by a resource (a person or a computer, for example). A business rule defines or constrains some aspects of the business, which must be respected during the whole process execution. Hence, the defined business rules control/limit the possibility of a stakeholder to choose an activity to execute.

As said, a declarative process makes use of a declarative language to express its business rules. ConDec is one declarative language, proposed by Pesic [17], used in the DECLARE modeling and execution tool. ConDec uses LTL (Linear Temporal Logic) formulae as basis to describe different constraint behaviors. The ConDec language is also used in this work to describe business rules in a declarative manner. The behaviors can be classified in the following templates:

- **existential templates** express the number of times an activity can or must be executed in the same process instance. This group has four templates: *exactly*(A, N) indicates that an activity A must be executed exactly N times; *existence*(A, N) indicates that an activity A must be executed at least N times; *absence*(A, N) indicates that an activity A must be executed at most N times; and *init*(A), which indicates the first activity to be executed in the process.
- **relational templates** express dependencies between activities. This group has five templates: *precedence*(A, B) indicates that an activity B cannot be executed before a certain activity A; *response*(A, B) indicates that, after the execution of an activity A, a certain activity B must also be executed; *succession*(A, B) corresponds to the conjunction of the templates response and precedence; *co-existence*(A, B) defines that if an activity A is ever executed, a certain activity B must also be executed and vice-versa; and *responded existence*(A, B) establishes that if an activity A is ever executed, activity B must also be executed (before or after the execution of A);
- **negation templates** refer to a negative version of the relational templates. For example, the template *not response*(A, B) indicates that after the execution of a certain activity A, activity B cannot be executed anymore (B cannot be executed after A) and *not co-existence*(A, B) indicates that after the execution of one of these two activities (A or B), the other one cannot be executed anymore.

3 REFlex

In this section, we describe how REFlex is inserted in the Business Process Management area. We associate REFlex with other imperative and declarative approaches, comparing and differentiating them. We also describe REFlex as an entire solution to define declarative business process, to control its execution and to orchestrate services as activities.

3.1 The BPM area

One of the steps in the BPM lifecycle is the modeling one. In this step the process models are defined according to the needs of the company. These models should have information about the flow of activities. To design such models we need a modeling notation able to support both technical and business users. The modeling notation must provide to the stakeholders the ability to communicate the models procedures.

These modeling notations must have well-defined semantics and formalization. A formal semantics helps to understand the defined business process and its execution. It is responsible for defining semantics models and the relations between them. In Figure 1 we can see some examples of modeling notations and their semantics. Considering imperative business processes, BPMN [26] and EPC (Event Process Chain [14]) are examples of modeling notations; their semantics are defined in CSP (Communicating Sequential Processes) [28, 27] or Petri nets [6] for BPMN and Petri nets [21] for EPC.

If some activities of the modeled business process are actually web services, we need to specify how these web services will be executed. For this purpose, an execution specification must be defined. It allows to describe how web services in a service-oriented architecture (SOA) interconnect and how they share data. For BPMN and EPC models, the BPEL language [10] can be used to the execution specification, which messages are typically used to invoke remote services, orchestrate process execution and manage events and exceptions.

After designing the process model and defining its execution specification at design time, an imperative business process is transformed into a workflow with orchestration information of the web services to be executed in runtime. The workflow describes each activity that must be performed during the execution of the business process, possible execution flows, the participants that execute each activity, and the information flow between them [29]. In addition to the workflow, the orchestration information represents the execution of the web services. These services can be composed and work in cooperation to enact the business process. These compositions allow creating new values from existing parts, higher level services, that share data to deliver the requested, or part of, solutions for the business process.

Regarding declarative processes, we can cite ConDec [17] as a modeling notation. ConDec is a language that describes a business process based on rule templates. It was proposed to be used primarily in DECLARE framework, which offers a graphical representation to each template. Hence, the user can define the

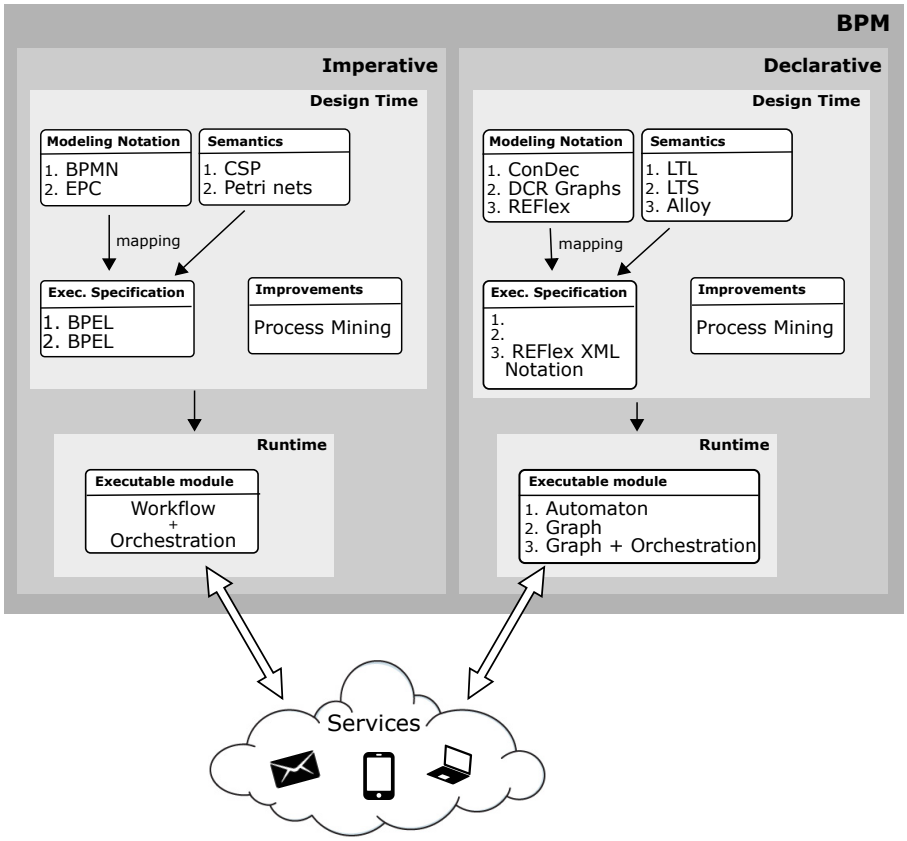

Fig. 1. Overview of BPM and REFlex.

business process graphically, which increases the understandability of the process. Each template in ConDec is mapped to a LTL formula, so that a ConDec model can be formalized in terms of LTL in a finite trace [12]. Because of the formalization, a runtime verification can be done focusing on the violations from the interference of multiple constraints. An automaton is generated to provide diagnosis about the business constraints during the execution of the model (at runtime) [7].

Another modeling notation for declarative business processes is the DCR Graphs [8]. In DCR Graphs a business process is specified in terms of activities and four different primitive constraints. Each constraint determines a relation between two activities. The entire business process constitutes a graph. DCR Graphs also provides a graphical tool to define a business process. The semantics of a DCR Graph is given as Labeled Transition System (LTS), which is a finite state when the graph is finite [8]. During runtime the graph controls the execution of the process, and the activities are labeled according to the current execution

step. The DCR Graphs primitives allow activities to be included and/or excluded to/from the graph during runtime.

Process mining is another subject of study in the business process area. A process mining technique aims at extracting, or discovering, a business process model from the event logs recorded by any information system. Process mining refers to methods for distilling a structured process description from a set of real executions [24], explaining the behavior observed in the event log. These techniques are most applicable when the business events are recorded in event logs but the information in these logs are not used to analyze the underlying process [22]. We consider process mining as improvement to the modeling notations and execution specifications made in design time, as it appears in Figure 1. Various tools and techniques in process mining have been developed for imperative models [24, 22, 23] and declarative ones [4, 13, 11].

3.2 REFlex as an entire solution to BPM

In this section, we will explain how REFlex works as entire solution to define declarative business processes. We will explain how a user can interact with REFlex. Using REFlex the user can also determine web services as activities, specifying their interactions and how they share data.

In REFlex, the definition of a business process (activities and rules) is done using the ConDec templates. In a previous work [20] we extended the definition of the business process to approach web service orchestration, besides the definition of activities and rules. We proposed an XML-based language capable of representing both the structure of the process (the REFlex graph) and the meta data associated to this structure (data types and web service connections). Through the XML file the user can specify variables (*int*, *float*, *double*, *String*, *boolean*, and *list*, which is an array of the other variable types) and service bindings. This latter determines which web service is linked to which activity, the location of their WSDL interface, and the binding, port type, and operation that should be called when the user chooses the activity to be executed. In [20] we describe how to write such XML.

Before starting the execution, the user must define the business process through the XML file, which contains all necessary information for its execution. This file will be parsed into a graph representation, with REFlex readable elements.

Through the user interface of a company tool that interacts with REFlex, the user is able to know: i) which activities are enabled to be executed at that moment; ii) the current values of global data shared by the activities (or web services), and iii) if the execution of the process can be finished at that moment or not. Figure 2 gives an overview of the interaction of users, the company tool and REFlex.

Once the user chooses an enabled activity to be executed, the company tool interacts with the REFlex orchestrator informing the chosen activity. If this activity is linked to a web service, the orchestrator will invoke the corresponding service operation. It uses the WSDL interface to automatically construct and

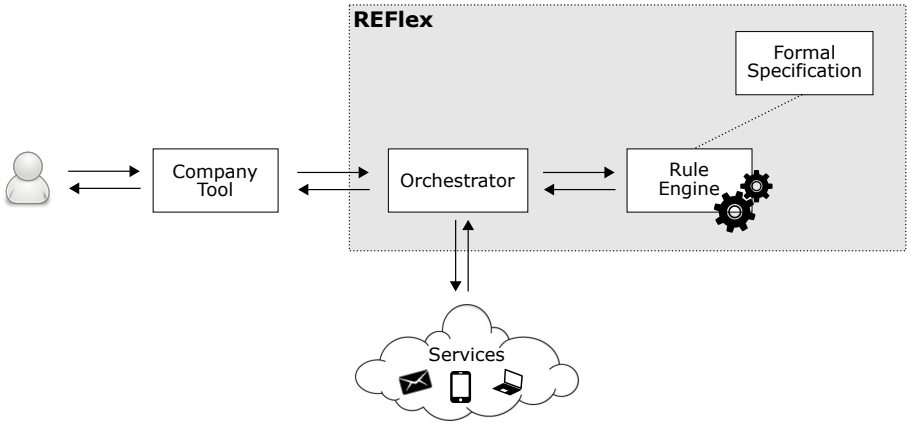

Fig. 2. Overview of the interaction of users, the company tool and REFlex.

interpret the messages that are sent/received to/from the web service. Otherwise, the orchestrator will wait the user to execute the activity manually.

After the execution of each activity, the rules need to be checked. To this end, REFlex orchestrator interacts with REFlex rule engine to update the status of the business rules. The communication between orchestrator and rule engine is given by events. Every time an activity is executed, the orchestrator generates a *DONE(activityName, data)* event and sends it to rule engine. The *activityName* represents the activity chosen by the user to be executed, and *data* is the current values of the data stored by the orchestrator.

In order to update the process instance status, the rule engine generates four kind of events to the orchestrator: *ENABLE(activityName)*, *DISABLE(activityName)*, *ENABLE_END()*, and *DISABLE_END()*. On one hand, events *ENABLE(activityName)* and *DISABLE(activityName)* update an activity state, allowing an activity to be marked as enabled (can be executed by the user) or disabled (cannot be executed by the user at that moment). On the other hand, events *ENABLE_END()* and *DISABLE_END()* informs if the process execution can be concluded or not, respectively. A process execution can be concluded if, and only if, there is no pending activity in the process. A pending activity is an activity that must be executed before the end of the execution. After receiving the events from the rule engine, the orchestrator can inform the company tool about the activities that became enabled and the ones that became disabled. Hence, the company tool can be updated, allowing the user to choose another, and at that moment enabled, activity.

The REFlex rule engine uses a graph to represent the current state of the process execution. It uses a *liveness-enforcing* mechanism to guarantee that unacceptable situations are never reached and to guide the user to not mistakenly drive the execution to such unacceptable situations [2]. It is also able to handle data conditions to restrict the existence of an activity or a rule during runtime.

Besides that, the semantics of the REFlex rule engine is formalized in the Alloy language [9], which relies on recent advances in SAT (Boolean Satisfiability) technology. The REFlex formalization [5] allows us to discuss the verification of the engine core properties. Through the core properties we can discuss the feasibility of the defined business process, and identify for example that there are contradictory rules in such business process. We can also discuss the *liveness-enforcing* mechanism properties and all the apparatus this mechanism makes use to infer new rules to the business process without interfering in the business process behavior.

4 Related Work

Our work lies within the scope of declarative BPs, in this section we discuss some important works in this area.

DECLARE is a rule engine system proposed by Pesic et al. [16] for modeling and executing declarative processes through an extensible graphical language called ConDec [17]. This language offers a set of graphical representations to describe control-flow rules that constrain the execution of process activities. DECLARE uses Linear Temporal Logic (LTL) as its formalism for the internal representation of business processes. Process enactment requires the construction of a Büchi automaton that contains all possible states of the process. This strategy leads to the well known problem of state space explosion, which limits the size and complexity of the business processes that can be executed using DECLARE.

Hildebrandt and Mukkamala [8] propose a graph-based model called DCR Graphs to specify business process rules. At runtime, DCR Graphs control the dynamic evolution of the process state. Since the states of the process are updated dynamically, this approach does not require a prior generation of all possible states. During the process execution, this graph expresses the runtime states. This work is similar to our work since it checks the rule compliance trough a graph structure. However, the DCR graphs only support two types of relation (condition and response relations) and two types of behaviors (dynamic inclusion and exclusion of events).

Declare Analyzer, proposed by Burattin [1], is a ProM plug-in that is able to measure the conformance of process execution logs to a set of business rules provided. However, it does not verify the rule compliance during the process execution. Instead, it uses the logs of the process execution to verify its conformance to the process rules.

5 Conclusions

We resumed and discussed some imperative notations to model business processes. For these models, the user can define an execution specification, which

associates activities of the business model to web services, allowing the automation of the process or part of. The disadvantage of such approach is that the user must determine all possible execution paths for the process execution. The orchestration of services that follow an imperative model is simple, given that the possible compositions are previously know.

On the other hand, declarative approaches emerged to provide flexibility to the business processes. The declarative processes are useful for companies whose business logic is constantly changing. A declarative model consists of a set of activities (actions) in a company and a set of business rules (constraints). Any activity is allowed to execute since its execution does not violate any of the defined constraints.

In a previous work [2], we proposed REFlex, a graph-based rule engine to execute declarative processes. In other work [20] we extended the definition of the business process to approach web service orchestration, besides the definition of activities and rules.

In this paper, we showed that REFlex can be considered as an entire solution to business process modeling. Different from other declarative approaches, using REFlex, an user has support to: i) model a business process; ii) trust in a formalized semantics that does not allow the user to be guided to an unacceptable situation; and iii) link (some) activities to web services, (semi-)automating the process execution.

As future work, we intend to apply REFlex in real companies that require flexible processes. This will give us the idea of the acceptance of the REFlex as a pure declarative approach. Other works [19] have already discussed the acceptance of declarative or hybrid approaches. We also intend to extend the work to provide any process mining technique. We believe that this could help the acceptance of REFlex, since the user would only modify/update a business model.

References

1. Andrea Burattin, Fabrizio Maria Maggi, Wil M. P. van der Aalst, and Alessandro Sperduti. Techniques for a posteriori analysis of declarative processes. In Chi-Hung Chi, Dragan Gasevic, and Willem-Jan van den Heuvel, editors, *EDOC*, pages 41–50. IEEE, 2012.
2. Renata M. de Carvalho, Natália C. Silva, Cesar A. L. Oliveira, and Ricardo M. Lima. Reflex: an efficient graph-based rule engine to execute declarative processes. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, 2013.
3. Renata M. de Carvalho, Natália C. Silva, Cesar A. L. Oliveira, and Ricardo M. Lima. A solution to the state space explosion problem in declarative business process modeling. In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering*, 2013.
4. Federico Chesani, Evelina Lamma, Paola Mello, Marco Montali, Fabrizio Riguzzi, and Sergio Storari. Exploiting inductive logic programming techniques for declarative process mining. In Kurt Jensen and WilM.P. van der Aalst, editors, *Trans-*

actions on Petri Nets and Other Models of Concurrency II, volume 5460 of *Lecture Notes in Computer Science*, pages 278–295. Springer Berlin Heidelberg, 2009.
5. Renata M. de Carvalho. *A graph-based model for declarative business processes*. PhD thesis, Center of Informatics, Federal University of Pernambuco, Recife, Brazil, 2015.
6. Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in {BPMN}. *Information and Software Technology*, 50(12):1281 – 1294, 2008.
7. D. Giannakopoulou and K. Havelund. Automata-based verification of temporal properties on running programs. In *Automated Software Engineering, 2001. (ASE 2001). Proceedings. 16th Annual International Conference on*, pages 412–416, Nov 2001.
8. Thomas T. Hildebrandt and Raghava Rao Mukkamala. Declarative event-based workflow as distributed dynamic condition response graphs. In *Proceedings Third Workshop on Programming Language Approaches to Concurrency and communication-cEntric Software, PLACES 2010, Paphos, Cyprus, 21st March 2010.*, pages 59–73, 2010.
9. Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.
10. Matjaz B Juric. *Ws-Bpel 2.0 for Soa Composite Applications with Oracle Soa Suite 11g*. Packt Publishing Ltd, 2010.
11. FabrizioM. Maggi, R.P.JagadeeshChandra Bose, and WilM.P. van der Aalst. Efficient discovery of understandable declarative process models from event logs. In Jolita Ralyt, Xavier Franch, Sjaak Brinkkemper, and Stanislaw Wrycza, editors, *Advanced Information Systems Engineering*, volume 7328 of *Lecture Notes in Computer Science*, pages 270–285. Springer Berlin Heidelberg, 2012.
12. FabrizioMaria Maggi, Michael Westergaard, Marco Montali, and WilM.P. van der Aalst. Runtime verification of ltl-based declarative process models. In Sarfraz Khurshid and Koushik Sen, editors, *Runtime Verification*, volume 7186 of *Lecture Notes in Computer Science*, pages 131–146. Springer Berlin Heidelberg, 2012.
13. F.M. Maggi, A.J. Mooij, and W.M.P. van der Aalst. User-guided discovery of declarative process models. In *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pages 192–199, April 2011.
14. Jan Mendling. Event-driven process chains (epc). In *Metrics for Process Models*, volume 6 of *Lecture Notes in Business Information Processing*, pages 17–57. Springer Berlin Heidelberg, 2008.
15. Selmin Nurcan. A survey on the flexibility requirements related to business processes and modeling artifacts. In *HICSS '08: Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, page 378, Washington, DC, USA, 2008. IEEE Computer Society.
16. M. Pesic, H. Schonenberg, and W.M.P. van der Aalst. Declare: Full support for loosely-structured processes. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, page 287, oct. 2007.
17. Maja Pesic. *Constraint-Based Workflow Management Systems: Shifting Control to Users*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2008.
18. Jan C. Recker, Michael Rosemann, Marta Indulska, and Peter Green. Business process modeling : a comparative analysis. *Journal of the Association for Information Systems*, 10:333–363, 2009.

19. HajoA. Reijers, Tijs Slaats, and Christian Stahl. Declarative modelingan academic dream or the future for bpm? In Florian Daniel, Jianmin Wang, and Barbara Weber, editors, *Business Process Management*, volume 8094 of *Lecture Notes in Computer Science*, pages 307–322. Springer Berlin Heidelberg, 2013.
20. Natália C. Silva, Renata M. de Carvalho, Cesar A. L. Oliveira, and Ricardo M. Lima. Integrating declarative processes and soa: A declarative web service orchestrator. In *Proceedings of the 2013 International Conference on Semantic Web and Web Services*, 2013.
21. W.M.P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639 – 650, 1999.
22. W.M.P. van der Aalst, H.T. de Beer, and B.F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, volume 3760 of *Lecture Notes in Computer Science*, pages 130–147. Springer Berlin Heidelberg, 2005.
23. W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713 – 732, 2007.
24. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237 – 267, 2003.
25. T. van Eijndhoven, M.E. Iacob, and M.L. Ponisio. Achieving business process flexibility with business rules. In *Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference, EDOC'08*, pages 95–104, Los Alamitos, September 2008. IEEE Computer Society Press.
26. Stephen A White. Introduction to bpmn. *IBM Cooperation*, pages 2008–029, 2004.
27. Peter Y.H. Wong and Jeremy Gibbons. Property specifications for workflow modelling. *Science of Computer Programming*, 76(10):942 – 967, 2011. Integrated Formal Methods (iFM09).
28. PeterY.H. Wong and Jeremy Gibbons. A process semantics for bpmn. In Shaoying Liu, Tom Maibaum, and Keijiro Araki, editors, *Formal Methods and Software Engineering*, volume 5256 of *Lecture Notes in Computer Science*, pages 355–374. Springer Berlin Heidelberg, 2008.
29. Michael zur Muehlen. *Workflow-Based Process Controlling : Foundation, Design, and Application of Workflow-driven Process Information Systems.* Logos Verlag, Berlin, 2002.