Challenges of Testing Business Process Models in Intra- and Inter-Organizational Context

Stefan Mijatov, Tanja Mayerhofer

Business Informatics Group
Vienna University of Technology, Austria
{mijatov, mayerhofer}@big.tuwien.ac.at

Abstract: In order to address the issue of complexity and changeability of business processes, as well as of the information technology used to implement these processes, business process models are used. Validating the functional correctness of such models is essential. Many approaches exist, that deal with validating the functional correctness of business process models in an *intra-organizational* context (i.e., the process is controlled and maintained by one business entity). However, today, business processes are usually carried out by several business partners, each providing its own services to accomplish more complex *inter-organizational* business processes. This induces new challenges for validating the functional correctness of business processes based on models. In this paper we provide an overview of existing approaches to validate business process models in both intra- and inter-organizational context and discuss arising challenges.

1 Introduction

In order to cope with complexity and changeability of business processes on one hand, and the information technology used to implement these processes on the other, *business process models* are created and maintained. These models can be defined at various levels of abstraction, from capturing the high level concepts of a business process (high level activities performed by different parties to realize a business process) to a low level specification of a business process (information and object manipulation activities). Furthermore, these models can be defined with different modeling languages, from standardized modeling languages such as BPMN [OMG11a], UML [OMG11b], and BPEL [OAS07], to proprietary modeling languages.

As business process models become central artifacts in business process management as well as in the development of information systems, ensuring their quality is of uttermost importance. The quality of a software artifact (e.g., a business process model) represents the correspondence of its realization to the specified requirements. These requirements can be functional (i.e., what the artifact should do) and non-functional (i.e., the level of performance that the artifact has to achieve). In this

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

paper we focus on techniques for validating the functional correctness of business processes based on models.

There are many challenges that have to be addressed in this context. On one hand, organizations use different modeling languages for specifying their business processes. For validating the functional correctness of models, precisely defined semantics of the used modeling languages is a prerequisite. However, many organizations use modeling languages without or with partially defined semantics. On the other hand, in today's highly globalized business world, business processes are often realized by several cooperating organizations in order to achieve a common business objective. In this context, new challenges arise, such as autonomy, heterogeneity, and the support for coordination, which all may induce additional challenges to validating business processes.

This paper is structured as follows. In Section 2 and Section 3 we present existing approaches for validating functional properties of business processes based on business process models in an intra- and inter-organizational context. Thereby, we consider analysis, simulation, verification, as well as testing techniques. We focus on the three most popular modeling languages for business process modeling, namely BPMN, UML activity diagrams, and BPEL. In Section 4, challenges arising for validating both intra- and inter-organizational business processes are discussed. In Section 5 we briefly describe our own testing framework for UML activity diagrams based on the fUML standard [OMG11c] and discuss how the identified challenges can be addressed by this framework. Finally, we conclude our paper in Section 6.

2 Validating Intra-Organizational Business Process Models

In this section we present relevant existing work on the validation of business processes modeled with the Business Process Model and Notation language [OMG11a], UML Activity Diagrams [OMG11b], and the Business Process Execution Language (BPEL) [OAS07], which represent the most popular modeling notations used in both academy and industry.

2.1 Business Process Model and Notation (BPMN)

One of the most popular modeling languages for specifying business processes at the level of domain analysis and high-level system design is the Business Process Model and Notation (BPMN) language [OMG11a]. BPMN models are composed of activity nodes denoting business events or tasks performed by people or software applications within the business process, and control nodes capturing the flow of control between activity nodes.

There exist many approaches on the functional validation of BPMN models. What is mutual to the findings of all these approaches is the lack of precisely defined

semantics of BPMN which is a prerequisite for the behavioral interpretation of BPMN models necessary for realizing their validation.

Dijkman *et al.* [DDO08] present an approach for statically analyzing business process models defined with BPMN. In this approach mapping rules are defined for transforming BPMN models into Petri Net models. The authors identified several issues with the BPMN specification, such as imprecisely defined semantics of executing a business process with several start events and proper process instance completion. The obtained Petri Net model is used as input for the ProM tool to check (1) the absence of dead tasks (i.e., tasks that cannot be executed) and (2) the proper completion of the business process model (i.e., the state of a process model in which an end event is reached and no other task is enabled).

Another similar approach is described by Raedts *et al.* [RPU+07]. Their approach is based on translating a BPMN model into an Extended Petri Net model and then translating the Extended Petri Net model into mCRL2, a process algebraic language, after which the mCRL2 toolset can be used. Based on this approach, the modeled business process can be statically analyzed, simulated, or verified, e.g., for analyzing its performance (e.g., process execution time) and verifying soundness. A process model is considered sound if each state in the Petri Net can be reached at least once in one of the possible executions, the final state of the Petri Net is reachable from any state of the possible executions (deadlock free), and for each execution of the Petri Net the final state is reachable (proper process completion) [vdA98].

A different approach is presented by Ligeza *et al.* [LKNP12]. Their approach is based on leveraging a rule-based system which is implemented in Prolog and composed of business rules and BPMN models specifying a business process to analyze previously defined correctness criteria, as well as the correctness of data flows. These criteria include the correctness of process components (tasks), data flows, splits, merge nodes, and finally of the overall process.

2.2 UML Activity Diagrams

The Unified Modeling Language (UML) [OMG11b] is a popular modeling language for specifying structural and behavioral aspects of a software system. The structural aspects describe the objects that exist in the system, as well as relationships among these objects. Behavioral aspects define both the history of interactions between objects over time, as well as the communication of objects to accomplish certain goals. UML contains many types of diagrams, such as class, object, sequence, activity, and state machine diagram, that enable a user to comprehensively specify all necessary details of a system, both on a high and a low level of abstraction. An investigation of the expressiveness and adequacy of the UML activity diagram notation for specifying business processes is presented by Dumas and ter Hofstede [DtH01]. They conclude, that despite some issues with the semantics specification and few missing modeling concepts, UML activity diagrams are expressive enough to be used for specifying business processes.

One approach that enables to validate the functional correctness of UML activity diagrams is presented by Staines [Sta08]. In this approach UML activity diagrams are translated into Petri Net models, which can subsequently be translated into Colored Petri Net models for which already existing analysis tools can be used to check properties, such as soundness.

Engels *et al.* [ESW07] propose another approach for verifying the correctness of UML activity diagrams. Their approach is based on applying Dynamic Meta Modeling (DMM) for defining the behavioral semantics of activity diagrams based on graph transformations. They apply the existing model checker tool GROOVE to check the soundness of the analyzed activity diagram.

Eshuis and Wieringa [EW04] present a tool for verifying activity diagrams which is an extension of the graph editing tool TCM. Upon specifying functional requirements and an activity diagram, both are translated into a transition system according to their formal semantics specification. This transition system serves as input for an existing model checker. In case that the requirements are not satisfied, a counter example is provided in the form of an execution trace that lead to the requirement violation.

All of these approaches specify the semantics of UML activity diagrams on their own, e.g., by translating them into Petri Nets. However, the semantics of a subset of UML has recently been precisely defined and standardized by the fUML standard [OMG11c] which specifies a virtual machine for executing UML activity diagrams. Based on the fUML virtual machine we elaborated a testing framework in previous work [MLMK13], which will be described in more detail in Section 5.

2.3 Web Services and Business Process Execution Language (BPEL)

In order to achieve interoperability between applications based on Web standards (e.g., SOAP, WSDL, UDDI), Web services are commonly used as implementation technology as they enable a loosely coupled integration of heterogeneous systems.

In order to define and execute a complex business process based on Web services, the Business Process Execution Language (BPEL) can be used. It provides a model and a grammar based on XML for describing collaborating business processes through which a complex business process is realized. It entails interaction through Web service interfaces and a logic for coordinating these interactions.

Mayer and Lübke [ML06] propose an architecture and implementation of a unit testing framework for validating processes defined with BPEL. The architecture of the framework is composed of four layers: test specification (i.e., how the test data and its behavior are specified), test organization (tests are organized into test cases, test suites, and test runs), test execution (test and process under test must be executed), and test result layer (results must be gathered, logged, and presented to the user). Possibly incorrect data received from the process under test is categorized into three types: incorrect content, no message returned, or an incorrect number of received messages.

Li and Sun [LS06] propose another implementation of a unit testing framework for BPEL, called BPELUnit. The basic idea of the framework is to transform process interactions through Web service invocations to class collaborations using method calls, and to then apply object-oriented testing techniques. Firstly, interface descriptions of the Web services under test are transformed into corresponding Java interfaces upon which mock objects are created that represent partner processes that are directly invoked by the process under test. A mock object for a process defines expected invocations and return values of the process and validates that invocations are performed with correct parameters during runtime. The authors identified several advantages of their framework: the testing process does not rely on the availability of partner processes, test case writing is simplified, the test case execution time is improved, and automatic regression testing is enabled.

Weber *et al.* [WHM10] suggest that soundness of business process models is a necessary but insufficient condition for correctness, and that it might me necessary to take into account also the effects of executing individual activities within the process. Their approach includes the verification of effect conflicts (consistency of effects of parallel activities), precondition conflicts (consistency of preconditions of activities), reachability (are there activities that are not reachable within possible executions), and executability (are there activities whose preconditions are false at the time they need to be executed).

3 Validating Inter-Organizational Business Process Models

To maintain a certain level of competitiveness, collaboration between partner companies is essential. In this context, companies focus on their specialized functions for which they have expertise, complementing their business processes with partners and suppliers. In such an environment, companies rely on inter-organizational business processes to realize their business. Bouchbout and Alimazighi [BA11] define inter-organizational business process as an organized group of joined activities carried out by two or more organizations to achieve a common business objective. The design and implementation of inter-organizational business processes open up new challenges that need to be addressed: the flexibility and ability to cope with change, decentralized management, peer-to-peer interactions, preservation of enterprise autonomy and privacy, and support for interoperability.

Breu *et al.* [BDE+13] point out several important aspects of inter-organizational processes: *distribution* (either one party is serving as a controller of the complete process, or no party is in control of the whole process), *behavior* (different languages for describing choreographies exist), *data* (data can be distributed between collaborating parties), and *resources* (quality of service and service-level agreements are necessary for establishing trust between parties concerning provided resources). The authors identify four main challenges of inter-organizational processes: flexibility, correctness, traceability, and scalability. *Flexibility* is defined as the variability of participating intra-organizational processes, looseness of their interactions, adapta-

tion as capability to adapt the process to emerging events, and evolution as ability to change the process according to changed business requirements. *Correctness* is a prerequisite of both intra- and inter-organizational processes, as it ensures non-disrupted functioning of a business process. *Traceability* is defined as the need to monitor which progress is made at which point in time for which steps of the process. And finally, *scalability* is necessary for ensuring the reliability of the process execution as the computing resource requirements change over time.

In Section 2 we gave an overview of existing approaches for validating intra-organizational business process models. However, for validating inter-organizational business processes, their peculiarities compared to intra-organizational business processes have to be taken into consideration. In the following, we provide an overview of existing approaches for validating the functional correctness of inter-organizational business processes.

Van der Aalst [vdA98] presents an approach for analyzing and verifying inter-organizational workflows. Each workflow of each participating partner is modeled as a Petri Net model and additional modeling concepts are used for defining the connections between communicating workflows. For each individual Petri Net model describing an intra-organizational workflow, the concept of *soundness* can be verified by using standard Petri Net techniques. The communication between partner processes in an inter-organizational process can be either synchronous or asynchronous. Synchronous communication is modeled as direct connection between transitions of Petri Nets defining the participating partner processes. Asynchronous communication is modeled as a connecting place between transitions of the Petri Nets defining the participating partner processes. A Petri Net model describing an inter-organizational business process is considered sound if it is both *locally* sound (each Petri Net model of each participating partner is sound) and *globally* sound (the complete Petri Net model of the inter-organizational process is sound).

Martens *et al.* [MMGF06] present a similar approach for analyzing and verifying the *compatibility* of BPEL processes. They define two dimensions of compatibility: *syntactical* compatibility (two BPEL processes can be composed only if the provided interfaces of those two processes are mutually compatible) and *behavioral* compatibility (the behavior provided by two processes must be compatible). The approach is again based on Petri Net models. BPEL processes are translated into Petri Net model representations which are then combined into a single Petri Net model. Thereof the soundness is checked for the combined Petri Net model.

Similarly, Dumas *et al.* [DBN08] discuss the notion of protocol compatibility between Web services and review a number of techniques for detecting incompatibilities and for synthesizing adapters for otherwise incompatible services.

Benatallah *et al.* [BT04] define different types of protocol compatibility, corresponding to different levels of service interoperability. They show how given two services and their specifications it is possible to formally determine their level of compatibility. In addition, they present how to identify similarities and differences between protocols, in order to asses their equality and replaceability.

Intra-Organizational Business Processes					
Authors	**Technique**	**Static analysis**	**Simulation**	**Verification**	**Testing**
BPMN					
Dijkman *et al.* [DDO08]	Translation into Petri Nets	×			
Readts *et al.* [RPU+07]	Translation into Extended Petri Nets and mCRL2	×	×	×	
Ligeza *et al.* [LKNP12]	Rule-based system		×	×	
UML AD					
Staines [Sta08]	Translation into Colored Petri Nets	×	×	×	
Engels *et al.* [ESW07]	Graph transformations		×	×	
Eshuis and Wieringa [EW04]	Translation into transition systems			×	
BPEL					
Mayer and Lübke [ML06]	Unit test framework based on XML and Xpath				×
Li and Sun [LS06]	Unit testing framework based on JUnit				×
Weber *et al.* [WHM10]	Verification method exploiting semantic annotations			×	
Inter-Organizational Business Processes					
PetriNet					
Van der Aalst [vdA98]	Composition of Petri Nets	×		×	
BPEL					
Martens *et al.* [MMGF06]	Translation into annotated Petri Nets	×		×	

Table 1: Overview of approaches for validating business process models.

Another approach for ensuring the correctness of a business process in an inter-organizational context is to analyze its compliance with so-called *reference models*. Reference models provide a set of generally accepted, sound, and efficient processes and can help to speed up and optimize the design of business process models as well as ease the compliance with regulations and requirements. An approach for measuring the compliance of processes with reference models is described by Gerke *et al.* [GCC09]. In their approach they define compliance as the degree to which a process model behaves in accordance with a reference model. Compliance is measured by validating the process instances of a model (i.e., possible executions) against the reference model by determining whether they are valid instances of the reference model or their degree of deviation.

A similar approach is provided by Weidlich *et al.* [WMPW11]. In this approach the behavioral consistency between a business process model specifying the system and a workflow model as implementation of that system is analyzed. Behavioral consistency of the two models is defined in terms of a causal behavioral profile, which represents a behavioral abstraction that includes dependencies in terms of order, exclusiveness, and causality between pairs of activities in the models. Besides the order of potential occurrences, also optionality and causality are described. Optionality of a transition is defined as the existence of a firing sequence leading from the initial to the final marking of the system which does not contain the transition. Causality is defined as the restriction that the transition can occur in the firing sequence only after another given transition. One application for causal behavioral profiles is to check the conformance of process logs captured for the execution of a system to the modeled process. This conformance is measured as to what degree the behavior of the log is captured in the respective model.

4 Challenges of Validating Business Process Models

In Section 2 and Section 3 we presented existing approaches for validating the functional correctness of intra- and inter-organizational business processes. An overview of the discussed approaches is provided in Table 1. Most of these approaches focus on analyzing the soundness of the modeled processes and are restricted to a specific modeling language—most of the modeling languages have no standardized formal behavioral semantics defined. Furthermore, there are many challenges in validating inter-organizational business processes, such as compositions of process models where some depending processes of business partners might only be available through contracts while their internals are not visible. Another challenge is the need for flexibility in terms of enabling the evolution of business processes without central control. All this leads to some still open challenges in validating inter-organizational business processes.

Another important challenge in validating business processes is the *variety of available modeling languages* for defining business processes and the lack of standardized behavioral semantics of these languages. Having a common language to specify the behavioral semantics of modeling languages might lead to standardized methods and tools for validating the correctness of business process models regardless of the used modeling language.

In inter-organizational business processes, several intra-organizational business processes of collaborating business partners are combined in order to accomplish a common business goal. If all of these intra-organizational business processes are defined by business process models and the connection between those business process models are defined, it is possible to combine them into one process model defining the complete inter-organizational process. Hence, based on this *combined business process model*, it is possible to validate the correctness of the inter-organizational process using the approaches for validating business process models presented in Section 2. This approach was followed by van der Aalst [vdA98] and Martens *et al.* [MMGF06] as described in Section 3.

However, one important challenge that has to be addressed when dealing with inter-organizational business processes is flexibility in terms of the *variability of participating intra-organizational processes*. Thus, cooperating partners might be unknown in advance and might change during the execution of an inter-organizational business process. In such a context, it is not realistic to assume the existence of a business process model defining the complete inter-organizational business process. In order to enable the validation of the correctness of each participating intra-organizational business process in this situation, the technique of *mocking* can be applied. In the realm of Web service testing, there exists some work on the automatic or semi-automatic generation of so-called mock services that imitate the services provided by partner organizations. The work done by Li and Sun [LS06] presented earlier is an example for such an approach. Furthermore, *reference models* and *behavioral profiles* can help to validate the correctness of inter-organizational business processes in the situation where no business process model for the overall

process is available. In this context, validating the conformance of the business processes to mock objects or reference models respectively behavioral profiles is required. Work done in this area by Gerke *et al.* [GCC09] and Weidlich *et al.* [WMPW11] was described earlier. However, there is still an open issue of implementing these approaches in different modeling context.

The challenge of flexibility of inter-organizational business processes also includes the fact that *business processes can change* over time. *Regression testing* can be an essential technique for validating the correctness of inter-organizational processes in the event of changes. In software testing, regression testing is the process of selectively re-testing the software system to ensure that new bugs have been fixed and no other previously working functions have failed (regressed) as a result of changes to the system. With the application of test automation as provided by unit testing frameworks, regression testing is eased as it enables to specify tests and re-run them automatically. In Section 2 we have discussed two unit testing frameworks for business process models developed by Mayer and Lübke [ML06], and Li and Sun [LS06]. However, advanced techniques available for regression testing of software systems, such as automatically selecting the test cases that have to be re-run after a change, measuring the test coverage, and automatically generating additionally required test cases, have not been addressed so far by testing approaches for business processes.

5 Testing UML Activities: A Framework Based on fUML

In order to provide a precise and complete specification of the behavioral semantics of a subset of UML, the fUML standard [OMG11c] was developed by OMG. The subset of UML addressed by the fUML standard consists of modeling concepts for defining the structure of a system with UML classes and the behavior of a system with UML activities. For defining activities, the fUML subset contains a set of predefined actions which can be used for defining object manipulations (objects and links between these objects can be created, destroyed, and modified) and communications between activities (activities can call other activities synchronously or asynchronously). Furthermore, modeling concepts for defining the flow of control and the flow of data between actions and activities are included in the fUML subset. The fUML standard defines a virtual machine that is capable of executing models compliant to this subset. This precise and complete semantics specification of UML activities enables the development of validation and verification methods and tools for business processes which are defined by UML activity diagrams.

In recent work [MLMK13], we elaborated a testing framework for validating the functional correctness of UML activity diagrams based on the virtual machine provided by the fUML standard[1]. An overview of our testing framework is depicted

[1]We provide an implementation of our framework integrated with the Eclipse Modeling Framework. For more information about the implementation, we refer the interested reader also to our project website:

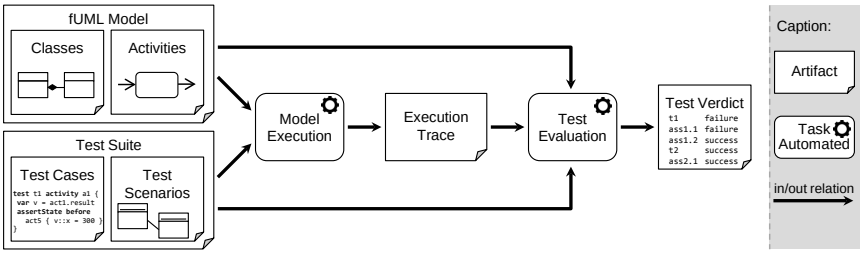

Figure 1: Testing framework for UML activity diagrams based on fUML.

in Figure 1. It consists of a *test specification language* and a *test interpreter*. The test specification language enables to define *test suites* composed of *test cases* defining assertions on the behavior of UML activities as well as *test scenarios* defining input values for the execution of the UML activities under test. Each test case in the test suite is evaluated by executing the activity under test with the input values defined by a selected test scenario using the fUML virtual machine. From the execution of the activity under test, an execution trace is obtained from the fUML virtual machine, which captures the runtime behavior of the executed activity. Based on this captured information, the assertions on the behavior of the activity defined by the test case are evaluated and the test verdict is calculated.

We support two kinds of assertions that can be defined on the behavior of a UML activity: *execution order assertions* for validating the execution order of nodes contained by the activity and *state assertions* for validating the state of objects modified by the activity.

Execution order assertions are specified by defining the order in which nodes contained by the activity under test have to be executed. Furthermore, the testing framework supports the assertion of the expected execution order of selected activity nodes only. Validating the correct execution order of nodes contained by an activity is an important capability when they are used to define the steps of business processes. However, this is not considered by the approaches discussed in Section 2. These approaches focus on the analysis of soundness properties which could also be done using our testing approach. In this respect, we are currently working on extending our testing framework to support parallelism in activity diagrams. For this, execution paths of an activity relevant to the specified execution order may be computed and can be checked against the specified order assertions. Furthermore, by analyzing the activity and the calculated execution paths, we could for instance identify dead tasks and validate the proper completion of an activity for a given input.

State assertions can be used to validate the state of an object at a specific point in time of the execution of the activity under test. To express the point in time at which the object shall be validated, a temporal quantifier, a temporal operator,

`http://www.modelexecution.org`.

and a reference activity node can be defined. Combining the temporal quantifier and the temporal operator it is possible to specify which states of the object should be checked after or before the execution of the reference activity node. A state assertion can contain several state expressions, each defining the expected value of one of the checked object's features. State assertions enable not only to validate the output of an activity execution, but also its intermediate results. This capability is usually not provided by unit testing frameworks, as they only enable to validate input-output relations.

Our framework enables to specify repeatable tests that can be automatically executed. Hence, it provides the necessary foundation for regression testing, which was identified as an enabling technique for supporting the evolution of inter-organizational business processes. However, it still remains to investigate techniques for test selection, measurement of test coverage, and the automated generation of test cases.

In order to support testing of inter-organizational business processes in the situation where each business partner has only access to his own intra-organizational business process, mocking the required partner processes would be valuable. In such case it is necessary to specify and analyze the correspondence of the partner process implementations to mock activities. For this, the already described approach by Weidlich *et al.* [WMPW11] could be used. We intend to investigate the possible application of this approach to UML activity diagrams.

6 Conclusion

As business process models become the main artifacts in business process management and information system development, validating their functional correctness becomes essential. In this paper we have presented several approaches for validating business process models in both intra- and inter-organizational context.

Most of these approaches are based on translating business process models into another formalism, such as Petri Nets, upon which standardized tools for analysis, simulation, and verification already exist. The reason for this is that the used modeling languages lack in providing formal execution semantics. Furthermore, most of the approaches concentrate on finding parts of a model that cannot be executed, as well as validating the proper execution completion of a model.

In the inter-organizational context, several business processes of collaborating partners are combined. Due to the needed flexibility of inter-organizational business processes, each partner might only have access to those parts of the process that they provide. In this situation it might be necessary to apply the techniques of mocking, where the processes provided by other partners are imitated, to enable the validation of the inter-organizational business process. The automatic or semi-automatic generation of such mocks might provide huge benefits. However, specifying behavioral contracts between business partners and checking the conformance of

implementations to those contracts are required.

The need for enabling the evolution of business processes constitutes another challenge in validating inter-organizational business processes which can be addressed by regression testing. However, advanced techniques, such as automated test case selection and generation do not exist so far.

In previous work we elaborated a framework for testing UML activity diagrams based on the fUML standard. Based on the identified challenges in validating business processes we are considering further extensions of our framework.

References

[BA11] K. Bouchbout and Z. Alimazighi. Inter-Organizational Business Processes Modelling Framework. In *Proc. of 15th East-European Conference on Advances in Databases and Information Systems (ADBIS'11)*, pages 45–54. CEUR-WS.org, 2011.

[BDE+13] R. Breu, S. Dustdar, J. Eder, C. Huemer, G. Kappel, J. Köpke, P. Langer, J. Mangler, J. Mendling, G. Neumann, S. Rinderle-Ma, S. Schulte, S. Sobering, and B. Weber. Towards Living Inter-organizational Processes. In *Proc. of 15th IEEE Conference on Business Informatics (CBI'13)*, pages 363–366. IEEE, 2013.

[BT04] B. Benatallah and F. Toumani. Analysis and Management of Web Service Protocols. In *Proc. of 23rd International Conference on Conceptual Modeling (ER'04)*, volume 3288 of *LNCS*, pages 524–541. Springer, 2004.

[DBN08] M. Dumas, B. Benatallah, and H. R. M. Nezhad. Web Service Protocols: Compatibility and Adaptation. *IEEE Data Engineering Bulletin*, 31(3):40–44, 2008.

[DDO08] R. M. Dijkman, M. Dumas, and C. Ouyang. Semantics and analysis of business process models in BPMN. *Information & Software Technology*, 50(12):1281–1294, 2008.

[DtH01] M. Dumas and A. H. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In *Proc. of 4th International Conference on the Unified Modeling Language (UML'01)*, volume 2185 of *LNCS*, pages 76–90. Springer, 2001.

[ESW07] G. Engels, C. Soltenborn, and H. Wehrheim. Analysis of UML Activities Using Dynamic Meta Modeling. In *Proc. of 9th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'07)*, volume 4468 of *LNCS*, pages 76–90. Springer, 2007.

[EW04] R. Eshuis and R. Wieringa. Tool Support for Verifying UML Activity Diagrams. *IEEE Transactions on Software Engineering*, 30(7):437–447, 2004.

[GCC09] K. Gerke, J. Cardoso, and A. Claus. Measuring the Compliance of Processes with Reference Models. In *Proc. of Confederated International Conference On the Move to Meaningful Internet Systems (OTM'09)*, volume 5870 of *LNCS*, pages 76–93. Springer, 2009.

[LKNP12] A. Ligeza, K. Kluza, G. J. Nalepa, and T. Potempa. AI Approach to Formal Analysis of BPMN Models. Towards a Logical Model for BPMN Diagrams. In *Proc. of Federated Conference on Computer Science and Information Systems (FedCSIS'12)*, pages 931–934, 2012.

[LS06] Z. J. Li and W. Sun. BPEL-Unit: JUnit for BPEL Processes. In *Proc. of 4th International Conference on Service-Oriented Computing (ICSOC'06)*, volume 4294 of *LNCS*, pages 415–426. Springer, 2006.

[ML06] P. Mayer and D. Lübke. Towards a BPEL unit testing framework. In *Proc. of 2006 Workshop on Testing, Analysis, and Verification of Web Services and Applications (TAV-WEB'06)*, pages 33–42. ACM, 2006.

[MLMK13] S. Mijatov, P. Langer, T. Mayerhofer, and G. Kappel. A Framework for Testing UML Activities Based on fUML. In *Proc. of 10th International Workshop on Model Driven Enginering, Verification and Validation (MoDeVVa'13)*, pages 1–10. CEUR-WS.org, 2013.

[MMGF06] A. Martens, S. Moser, A. Gerhardt, and K. Funk. Analyzing Compatibility of BPEL Processes. In *Proc. of Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW'06)*, pages 147–154. IEEE, 2006.

[OAS07] OASIS. Web Services Business Process Execution Language (WS-BPEL), Version 2.0, April 2007. Available at: `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[OMG11a] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011. Available at: `http://www.omg.org/spec/BPMN/2.0/`.

[OMG11b] OMG. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, August 2011. Available at: `http://www.omg.org/spec/UML/2.4.1`.

[OMG11c] OMG. Semantics of a Foundational Subset for Executable UML Models (fUML), Version 1.0, February 2011. Available at: `http://www.omg.org/spec/FUML/1.0`.

[RPU+07] I. Raedts, M. Petkovic, Y. S. Usenko, J. M. E. M. van der Werf, J. F. Groote, and L. J. Somers. Transformation of BPMN Models for Behaviour Analysis. In *Proc. of 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS'07)*, pages 126–137. INSTICC Press, 2007.

[Sta08] T. S. Staines. Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets. In *Proc. of 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'08)*, pages 191–200. IEEE, 2008.

[vdA98] W. M. P. van der Aalst. Modeling and Analyzing Interorganizational Workflows. In *Proc. of 1st International Conference on Application of Concurrency to System Design (ACSD'98)*, pages 262–272. IEEE, 1998.

[WHM10] I. Weber, J. Hoffmann, and J. Mendling. Beyond soundness: on the verification of semantic business process models. *Distributed and Parallel Databases*, 27(3):271–343, 2010.

[WMPW11] M. Weidlich, J. Mendling, A. Polyvyanyy, and M. Weske. Causal Behavioral Profiles - Efficient Computation, Applications, and Evaluation. *Fundamenta Informaticae*, 113(3-4):1009–1025, 2011.