Towards a Foundational Framework for Developing and Testing Inter-organizational Business Processes

Philip Langer
Vienna University of Technology
Vienna, Austria
langer@big.tuwien.ac.at

Stefan Sobernig, Gustaf Neumann
Vienna University of Economics and Business
Vienna, Austria
stefan.sobernig@wu.ac.at
gustaf.neumann@wu.ac.at

Abstract: Modeling and analyzing inter-organizational business processes is complicated substantially by heterogeneous process-modeling languages (e.g., surface vs. analysis languages) as well as by their inherent properties of loose coupling and local control (views). On top, multiple concerns must be addressed when modeling inter-organizational business processes, such as process data, behavior, distribution, and resources management. We believe that more research effort is needed towards establishing a language-oriented foundation for modeling inter-organizational business processes. This paper lays out one direction of language engineering towards this goal, collects generic and specific requirements on a language framework for IOPs, and discusses them tentatively.

1 Introduction

As a result of the specialization and globalization of businesses in a world, in which political boarders disappear gradually, cooperation among multiple organizations is increasingly important. Certainly, this trend has a major impact on the development, implementation, and maintenance activities of business processes. It is often insufficient to consider only the internal processes of an organization in isolation. Instead, techniques for ensuring an efficient interplay of multiple processes across organizational boundaries and for maintaining correctness and conformance of the involved processes, while ensuring flexibility in adapting and replacing certain services on the fly, are increasingly required.

The design, enactment, and analysis of inter-organizational business processes (IOPs) has attracted substantial research, under the umbrella of inter-organizational workflows [51], inter-enterprise business processes [8], and process-driven SOA [23]. The field, however, still poses major challenges [5]. As different organizations may use different languages and even different paradigms to specify processes [47], we face the challenge of *heterogeneity* in the involved process models hampering a uniform analysis of the processes' interplay. In an inter-organizational setting, with *loose coupling and no control about partners*, external processes may change unnoticed and partners need to be replaced frequently. This implies continuous checking of the correctness and the conformance of the IOP. To enable the efficient development and maintenance of IOPs, as well as to ensure interoperability among the involved processes, all concerns, ranging from data, behavior,

distribution through resources [5] should be treated as a conceptual whole. It remains an open challenge to come up with a *unified, holistic engineering approach* which takes all concerns of IOPs, as well as the interplay of concerns, into account.

In this paper, we put our initial ideas about a unified, foundational modeling framework for IOPs and about critical design requirements on such a framework up for discussion. The proposed framework aims to address both the heterogeneity of concepts, languages, and methods for designing, developing, and testing IOPs *and* the integration of IOP concerns, while at the same time enabling the efficient separated development of distinct concerns with tailored languages. It is evident that such a framework cannot be established without the help and the feedback of the broader research community, including the MinoPro committee and the MinoPro participants. To this end, this paper should act as a stimulus for joint research and critical discussion.

In Section 2, we first elaborate on the language-oriented engineering process that we plan to apply in our work towards the foundational modeling framework. To gather a first set of IOP-specific requirements on both the engineering process and the resulting language artifacts, we provide a selective survey of existing work in designing, developing, and testing IOPs in Section 3. This way, we outline how we plan to aggregate the current state of the art in IOP languages, concepts, semantics, and formal techniques by identifying commonalities and variations. Based on these aggregated findings, we then briefly discuss cornerstones of a foundational IOP modeling framework in Section 4.

2 Language Engineering for Integrated IOP Modelling

A *model* of an IOP is a precise and concise definition of the intended characteristics of a business process. An IOP model is expressed in a formally defined software language; that is, a language having a mathematical definition of both its abstract syntax and its semantics. Therefore, the model becomes amenable to verification and validation. In addition, an IOP model is executable [20] to allow for immediate behavioral inspection (e.g., in a simulated environment) or to actually enact a process (e.g., by instrumenting a process-driven, service-based software system). An IOP model covers either one or even multiple process concerns at the same time (data, behavior, distribution, resources; see also Section 3). To allow for concern-specific, partial IOP models or for concern-specific viewpoints on an integrated IOP model, a language-based IOP framework is realized as a family of modeling languages. A *software-language family* [58] is an infrastructure to create a number of integrated, tailored software languages based on a number of shared language-implementation assets. These can be used to define tailored abstract syntaxes, tailored concrete syntaxes, as well as derived semantics specifications, based on a common core of language abstractions and semantics. An IOP modeling language can be tailored towards an IOP concern (concern-specific view, in short; [5]) and/or towards a domain-specific process view [8].

We plan to follow documented procedural guidelines for domain-specific software-language engineering ([19, 42]; see also Figure 1). Early in the engineering process, the *scope* for and the *purpose* of the IOP language framework must be established. This involves systematic reviews and a systematic mapping of existing IOP research, in particular lan-

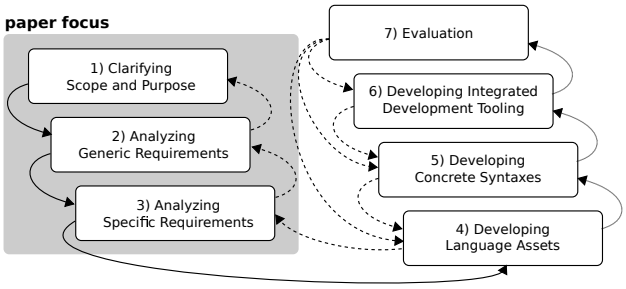

Figure 1: Overview of language-family engineering steps based on [19, 42]

guage projects and language-driven development tasks (process enactment, verification, validation). In addition, the availability of reusable IOP language artifacts and profiles of IOP stakeholders should be investigated. This step is particularly critical because the framework should incorporate and integrate with existing IOP languages. Collecting and eliciting *generic requirements* involves consulting collections of documented software-language design rationale (e.g., architectural patterns, language-implementation patterns, workflow patterns) as well as decisions on the subsequent engineering steps (e.g., extraction-based vs. mockup-based language development; [42]). The evaluation of different language-implementation strategies (e.g., metamodelling, grammars, language embedding) and of different concrete-syntax types (e.g., diagrammatic vs. textual) also fall into this category. Realizing the *semantics* of modeling languages may either be specified by mapping the language onto an existing semantic domain (such as petrinets) or by weaving the semantics into the metamodel (operational semantics [39, 35]) using a language for which the semantics are already available [6]. Then, there are *IOP-specific requirements* to be documented using a precise representation. For example, usage scenarios, concrete-syntax mock-ups etc. can be extracted from the systematic map of language projects drawn up in a previous step. In addition, domain-specific sources are consulted, such as material on workflow patterns [50]. Principles of view-based process modeling, referring to concern-specific and stakeholder-specific process views, will be screened [45, 46, 44].

Specifying *language-implementation assets* comprises the construction of a core abstract syntax and semantics, as well as syntax and semantics extensions covering the variable language parts. This is then complemented by realizing one or several *concrete syntaxes*. Finally, repeated empirical *evaluations*, for instance, based on the initially collected scenarios, are conducted and the results are fed back into re-designing iterations. The present paper contributes to a first structuring of relevant IOP sub-domains in preparation of the first three steps (scoping and requirements gathering; see Figure 1).

3 State of the Art in Developing and Testing IOPs

There is a voluminous body of material on software languages for IOP development and on corresponding techniques of verification and validation of business-process models. In the following, we highlight selected contributions (e.g., by identifying a single flagship pub-

lication per technique) to identify the underlying concepts and formal techniques, rather than providing a comprehensive survey, within the page limits set for this paper.

3.1 Data

The concern of data addresses the structural and semantical aspects of messages that are exchanged among participants of the IOP. As IOPs might not have one party serving as central control point, also the data can be distributed and heterogeneous posing the challenge of transactions and semantic integration, respectively.

Languages. The approaches to modeling of data structures have converged over past decades. Here, we can distinguish among relational approaches, such as the relational model by Codd [10] and the Entity-Relationship Model by Chen [9], tree-oriented approaches, such as XML Schema, object-oriented approaches, such as UML class diagrams, graph-oriented approaches, such as RDF and the introduction of ontologies with OWL, and document-oriented approaches [26], such as UN/CEFACT Core Components. Although they are based on different paradigms, there has been a consensus and a clear understanding about these paradigms, which lead to well-established approaches to translate among them without loosing information (e.g., object-relational mapping and JAXB[1]).

Formal techniques. To address the challenges of data distribution and distributed data integrity, transaction models of databases have been adapted to the workflow context [1] and web services [41]. With respect to heterogeneity and interoperability, for instance, Deutsch et al. [14] proposed a verification technique of exchanged data for data-centric processes. In case the exchanged artifacts are incompatible—requiring data transformations—semantic annotations [52] were applied successfully. To cope with evolution of data structures, semantic annotation paths were proposed for enriching WSDL interfaces [28]. Moreover, Weber et al. [53] combine the control-flow verification with checking pre-conditions on data items expressed in semantic process descriptions (OWL-S and WSMO).

3.2 Behavior

IOP behavior refers to control- and data-flow aspects (at design time), as well as possible runtime states during process execution. Key properties in this concern are correctness (e.g., soundness, liveness), as well as process conformance with business rules.

Languages. When it comes to modeling the behavior of business processes, the landscape of paradigms and languages is more diverse and little consensus exists on fundamental concepts [47]; especially when looking at language adoption in industry. The applied languages in modeling of business process mainly depend on the underlying purpose of the IOP model. Modeling processes with the aim of conceptually defining them is mainly done using semi-formal languages, such as BPMN, EPCs, and UML activities. Recently, declarative approaches such as DECLARE [49], gained momentum for specifying loosely structured processes. Rather than defining the control flow explicitly, declarative process models specify a list of activities and constraints regarding the execution order of those activities, usually captured using linear temporal logic (LTL) expressions. If the purpose of an IOP model is to analyze and to verify IOPs, rather modeling them conceptually

[1] `https://jaxb.java.net`

only, *formal languages* are employed, such as petrinets, finite state automata, and process calculi. Whereas both conceptual and formal process models abstract from implementation detail, *execution languages*, such as BPEL, aim at capturing enough detail to enable the deployment and the enactment of the modeled processes. Note that formalizations of conceptual modeling languages and execution languages exist [40], however, these formalizations are often incomplete and ignore ambiguities in the source languages.

Formal techniques. The verification of correctness of business processes has been intensively studied for well-defined correctness properties [56]. Most of this work applies control-flow analysis based on process models conforming to a clear token-flow semantics, such as petrinets, workflow nets, and YAWL. These formal concepts and verification methods have also been applied successfully to execution languages, such as BPEL, and more conceptual languages, such as UML activities. This, however, implies limiting the support to a subset of the languages' concepts or abstracting from ambiguities stemming from the informally defined semantics of conceptual languages. Design-time verification methods have also been developed for declarative process models to check whether, for instance, a so-defined process contains dead activities (i.e., there is no valid execution order containing the respective activity) or conflicting constraints (i.e., there is no valid execution order at all; [49]). These methods have later been adopted also for run-time verification [33].

Besides existing work on process verification, several researchers also addressed the validation of business processes according to certain functional requirements. Functional requirements are mostly specified using expected execution orders [59] or conformance relationships to reference models, which are checked based on graph comparisons [15], on execution traces [22], on causal dependencies of activities [55], and on equivalence notions using model checking [31]. Besides, there is a unit-testing framework for UML activities [37], which considers alongside execution orders, also the object states. Similarly, a unit-testing framework for BPEL processes [34] enables to specify and to check assertions on messages. Furthermore, methods from the domain of program analysis have been adopted to process models, such as symbolic execution [3] and test-case generation [57].

3.3 Distribution

One of the most obvious and, at the same time, urging concerns of IOPs is distribution, which includes the coordination and choreography of organizations participating in IOPs, as well as testing of the interoperability and conformance of interacting processes.

Languages. For modeling the distribution of control, two approaches are applied predominantly [13]. Using *interaction models*, the choreography of processes is defined in terms of a workflow containing activities that represent the message exchange among participating partners (e.g., WS-CDL). As the interaction is specified in terms of a process, choreographies may be specified using the same formalisms as used for specifying the behavior of the processes themselves. In contrast, with *interconnected interfaces modeling*, the control flow is defined per participant, i.e., the individual interface behavior models are stitched together using message links. Approaches to bridge between the two schemes have been proposed [36].

Formal techniques. From the perspective of distribution of IOPs, most of the work is concerned with validating the functional conformance and interoperability with respect to the behavior of the interacting processes. The conformance requirements can be specified using reference models, contracts [48] or, more generally, conformance rules turning into assertion checks on the execution order and, potentially, on the system state. Therefore, validation methods for checking the conformance of *intra*-organizational processes have been adopted and extended also for validating conformance and interoperability of IOPs, such as checking structural and semantic conformance [32], using the notion of causal dependencies of activities [55], or applying model checking to verify certain conformance rules or correctness properties [27].

3.4 Resources

Key to IOPs are means of adaptive resources management. Resources include computational, network, storage, and human resources required to carry out technical and non-technical tasks set by a business process. From a resources perspective, a business process translates into a set of precedence-related activities, which are to be executed on a set of resources. In an inter-organizational setting, however, processes are required to run under a variable, unbounded set of resources. Therefore, achieving on-demand (*elastic*) re- and de-allocation of resources to tasks has been identified as a key challenge [5, 18].

Languages. The notions of different types of resources (e.g., network, computation, human) and different types of elementary resource management activities, such as monitoring, scheduling, allocating, must be made explicit in IOP models. Attempts to integrate business-process modeling with resources modeling at the language level are still rare and in an early stage. Tai et al. [43] put forth the conceptual notion of infrastructure units which extends to human process resources (a.k.a. *social compute units* as in [17]). More recently, Janiesch et al. [] have proposed a set-theoretic metamodel to model key deployment abstractions of business processes (e.g., entities such as services, virtual machines, processes, and tasks as well as their relations in terms of service or task dependencies) on distributed resources infrastructures. Previous approaches on adaptive resource management build on similar conceptual metamodels, e.g., to express their reactive algorithms and their predictive statistical heuristics [24]. Alternatives are formal metamodeling techniques, such as an EMOF metamodel and auxiliary semantics expressed in OCL [7]. For predictive scheduling, colored petrinets have also been proposed to model dependency structures between tasks [2].

Resources modeling borrows many abstractions from languages for modeling software-system policies [12], including service-level agreements [24], and for modeling (business) rules and rules interchange [38]. Languages for defining the various deployment descriptors in business-process execution as well as for software services, distributed software components, and distributed computation containers (e.g., cloud nodes, virtual machines) are the second point of reference. The latter include extensions to description languages such as OVF [7, 11]. To this end, the resources concern strongly relates to the IOP distribution concern (see above). A second linkage between the two concerns are automated resource management approaches building, for instance, on distributed service-based QoS monitors [24] and on agent-based resources (re-) negotiation [54].

Formal techniques. In the business-process context, work on adaptive resources management and formal resources modeling addresses primarily optimality properties of resource-aware business processes (e.g., cost and/or time optimality [4]) from the perspective of different process stakeholders (e.g., customer, process participant, IaaS provider) and for different scopes (e.g., for a single and across multiple processes or process instances). Second, identifying SLA violations ahead of time is an open research issue. A first set of works relies mainly on constructing, evaluating, and calibrating *statistical prediction systems* based on monitoring data. Leitner et al. [30], for example, devise an approach to create and to train statistical prediction models (e.g., decision trees, neural networks, and auto-regression models) based on process monitoring data to deliver forecasts on SLOs such as delivery time, service availability etc. in service-based systems. Another family of approaches employs *online-testing techniques* to establish whether SLA/SLOs are expected to be violated and to trigger adequate resources adaptations. Online testing involves auxiliary, model-based test-case generation and selection techniques [16]. Ivanović et al. [25], on the contrary, apply an analytical approach based on deriving and solving *constraint sets* over atomic QoS probes (e.g., available for single services) and orchestration structure to predict upper and lower bounds for the expected, orchestrated QoS. A third challenge is correctness checking of adaptation (elasticity) operations [18]. Amziani et al. [2] employ an equivalence method on condensed state spaces of *colored petrinets* to proof that the adaptation operations proposed (resource duplication and removal) do not have unwanted side effects during process enactment (e.g., an increase in invocations). In addition, the authors propose model checking on *reachability graphs*, derived from colored petrinets representing resource-aware processes, to establish whether important properties hold under certain adaptation operations, including QoS violation by exceeding maximum capacities, deadlocks during call transfers, and adaptation loops.

4 Sketching out Language Foundations for IOPs

Ensuing from the body of existing concepts, languages, and methods across all four crucial IOP concerns, we aim to clarify the scope of the foundational framework, elicit important generic and specific requirements (see Section 2), and discuss initial ideas towards the foundational core for IOPs.

Generic Requirements. The language-oriented, foundational IOP framework must balance between two opposing but closely related forces. To provide effective modeling support, on the one hand, the framework must enable the development for distinct IOP concerns using tailored languages at different levels of abstraction (e.g., design vs. analysis languages). On the other hand, to allow for global IOP testing, partial concern-specific view models of an IOP must be integrated at some point and must be kept consistent during model co-evolution. Due to the magnitude of existing concepts, languages, and methods, another highly critical generic requirements is to accomplish a minimal foundational core language that enables a convenient integration of existing work and that provides an inherent extension mechanism to meet future requirements in IOP research. Therefore, this mechanism should allow for building syntactic and semantic extensions by instantiating and reassembling concepts and semantics of the foundational core language.

IOP-specific requirements. In Section 3, we highlighted flagship contributions on fundamental concepts and formal techniques from the otherwise extensive body of existing work on IOPs. Key findings are that relevant *structural concepts* include document-oriented or object-oriented data structures, constraints on data structures, through to concepts from knowledge representation, such as description logics. Based on these structural formalisms, conformance checking between messages and process/service interfaces or contracts is performed. Additionally, inference mechanisms for semantic alignment of heterogeneous data structures in messages becomes available.

Behavioral concepts of IOPs are described at different abstraction levels. This includes the notions of opaque or precisely specified activities and their execution dependencies, either using an explicit control-flow specification or temporal logics. Regarding their semantic expressiveness, for many analytical applications, both can be seen as equivalent. For instance, in both approaches a labeled transition system (cf. reachability graph of petrinets) can be computed to represent the state space. However, in temporal logics there is no explicit notion of concurrency, although nondeterminism can be modeled. Nevertheless, explicit concurrency modeling—such as enabled by petrinets, actor models, and process calculi—can be paramount. Besides, modeling synchronous and asynchronous message passing through channels is crucial, especially for choreographies using interconnected interfaces modeling. Other behavioral paradigms being used are event-driven process design, building on event-condition-action (ECA) rules, and state charts.

As for *formal techniques*, petrinets and declarative workflow models are subjected to *control- and data-flow analyses* extensively. By computing a concrete or symbolic state space from IOP models, e.g., in terms of a labeled transition system, also *model checking* techniques have been proven useful to verify business rules and IOP conformance rules. Since process enactment can be simulated, based on paths and path conditions, another line of research applies techniques from program analysis to business processes, such as symbolic execution and test-case generation. Besides these behavioral formalisms, we also identified *stochastic and non-functional concepts* being introduced to process models, such as time and computing resources. These formalisms allow to run predictive analyses and simulations regarding SLAs, quality of service, and resources management.

Towards a Foundational Core for IOPs. In next steps, the specific and generic requirements identified above will guide us towards condensing existing languages, concepts, and semantics into a common foundational core for IOPs. The objectives are that existing formal verification techniques and validation techniques should be supported as is, while also allowing for combining these techniques in novel ways and across IOP concerns. Please also note that the foundational core must not necessarily correspond to popular surface languages (i.e., the languages currently adopted IOP stakeholders). More importantly, the core should focus on key concepts and semantics, capable of reflecting and realizing the semantics of several surface languages *across* IOP concerns (e.g., ECA semantics in the distribution and resources concern). To accommodate changing requirements, a minimal but extensible subset of language abstractions is clearly preferable over a union of all available concepts and languages, which is why we envision to adopt principles and techniques from (model-driven) language engineering for providing built-in extension points regarding abstract syntax, semantics, and concrete syntax to address different IOP concerns.

In a model-driven approach, the foundational core of IOPs could build on a small object-oriented structure modeling language, such as EMOF or a subset of the UML class meta-model. For behavioral concerns, given the large body of research on verification and correctness of workflow nets, we plan to evaluate the adoption of compatible control-flow and object-flow semantics. This way, deriving labeled transition systems from process models as a basis for model checking becomes possible. In addition, modeling facilities for synchronous and asynchronous message flows will be considered. Interactions modeling between different process partners (and partner-specific process views) across organizational boundaries will require particular structuring concepts in behavioral models. At the same time, the manipulation of objects expressed in the structural language core should be supported (e.g., to allow for representing and for reasoning about state changes in precisely specified processes). For simulation-based analyses, IOP models should have the ability to represent runtime information; therefore, the foundational core should be capable of capturing execution states, events, and traces, which in turn would enable symbolic execution and test-case generation.

A potential candidate for adoption as one languages in the IOP core is the recently standardized foundational UML[2] (fUML), which consists of a subset of activities and classes with formally specified execution semantics. fUML adopts the token-flow semantics of petrinets and supports—besides object manipulation—also concurrency and signals for asynchronous message passing.

For providing *syntactic extensions* of languages (e.g., to add stochastic distributions to edges), the foundational core should provide lightweight extension mechanisms [29], as known from UML profiles [21]. The *extensibility* of core semantics in modeling languages is still an open research topic [6]. One idea is to introduce *semantic profiles*; that is, profiles for which an operational or translational semantics, again based on the foundational core for IOPs, can be specified. With such semantic profiles (e.g., one for event-driven behaviors), they can be applied on top of the core semantics by injection.

5 Concluding Remarks

With this paper, we present initial ideas on a language-engineering approach to construct a foundational modeling framework for designing, analysing, and testing inter-organizational processes. A resulting framework is foundational in the sense of enabling designing, developing, and testing IOPs in a unified manner, across the boundaries of interdependent IOP concerns. In addition, following this language-engineering procedure, we set the scope and identified first requirements for such a framework based on a mapping of existing concepts and methods available for BPM.

We see this paper as a first step towards building and establishing such a framework. A key challenge is designing the framework at the sweet spot of expressiveness, of facilitating IOP analyzability, and of equipping the framework with extensibility to meet future requirements in IOP research. We kindly invite the readers and the wider research communities on modelling inter-organizational processes to provide feedback and to join forces on a community-driven effort to tackle this challenge.

[2] fUML; `http://www.omg.org/spec/FUML/1.0`

References

[1] G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Günthör, and C. Mohan. Advanced transaction models in workflow contexts. In *Proc. ICDE*, pp. 574–581. IEEE, 1996.

[2] M. Amziani, T. Melliti, and S. Tata. Formal modeling and evaluation of stateful service-based business process elasticity in the cloud. In *Proc. OTM*, volume 8185 of *LNCS*, pp. 21–38. Springer, 2013.

[3] L. Bentakouk, P. Poizat, and F. Zaïdi. A formal framework for service orchestration testing based on symbolic transition systems. In *Proc. TESTCOM*, volume 5826 of *LNCS*, pp. 16–32. Springer, 2009.

[4] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan. Resources allocation and scheduling approaches for business process applications in cloud contexts. In *Proc. CloudCom*, pp. 496–503. IEEE, 2012.

[5] 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. Sobernig, and B. Weber. Towards living inter-organizational processes. In *Proc. CBI*. IEEE, 2013.

[6] B. R. Bryant, J. Gray, M. Mernik, P. J. Clarke, R. B. France, and G. Karsai. Challenges and directions in formalizing the semantics of modeling languages. *Computer Science and Information Systems*, 8(2):225–253, 2011.

[7] C. Chapman, W. Emmerich, F. Marquez, S. Clayman, and A. Galis. Elastic service definition in computational clouds. In *Workshop Proc. NOMS*, pp. 327–334. IEEE, 2010.

[8] I. Chebbi, S. Dustdar, and S. Tata. The view-based approach to dynamic inter-organizational workflow cooperation. *Data & Knowledge Engineering*, 56(2):139–173, 2006.

[9] P. P.-S. Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[10] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[11] G. Copil, D. Moldovan, H. L. Truong, and S. Dustdar. Sybl: An extensible language for controlling elasticity in cloud applications. In *Proc. CCGrid*, pp. 112–119. IEEE, 2013.

[12] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In *Workshop Proc. POLICY*, volume 1995 of *LNCS*, pp. 18–38. Springer, 2001.

[13] G. Decker, O. Kopp, and A. Barros. An introduction to service choreographies. *Information Technology*, 50(2):122–127, 2008.

[14] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. ICDT*, pp. 252–267. ACM, 2009.

[15] R. Dijkman, M. Dumas, B. Van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2):498–516, 2011.

[16] D. Dranidis, A. Metzger, and D. Kourtesis. Enabling proactive adaptation through just-in-time testing of conversational services. In *Proc. ServiceWave*, volume 6481 of *LNCS*, pp. 63–75. Springer, 2010.

[17] S. Dustdar and K. Bhattacharya. The social compute unit. *IEEE Internet Computing*, 15(3):64–69, 2011.

[18] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong. Principles of elastic processes. *IEEE Internet Computing*, 15(5):66–71, 2011.

[19] U. Frank. Domain-specific modeling languages: Requirements analysis and design guidelines. In *Domain Engineering*, pp. 133–157. Springer, 2013.

[20] N. Fuchs. Specifications are (preferably) executable. *Software Engineering Journal*, 7(5):323–334, 1992.

[21] L. Fuentes-Fernández and A. Vallecillo-Moreno. An introduction to UML profiles. *UPGRADE*, V(2):6–13, 2004.

[22] K. Gerke, J. Cardoso, and A. Claus. Measuring the compliance of processes with reference models. In *Proc. OTM*, volume 5870 of *LNCS*, pp. 76–93. Springer, 2009.

[23] C. Hentrich and U. Zdun. *Process-Driven SOA: Patterns for Aligning Business and IT*. Infosys Press, 2012.

[24] P. Hoenisch, S. Schulte, and S. Dustdar. Workflow scheduling and resource allocation for cloud-based execution of elastic processes. In *Proc. SOCA*, pp. 1–8. IEEE, 2013.

[25] D. Ivanović, M. Carro, and M. Hermenegildo. Constraint-based runtime prediction of SLA violations in service orchestrations. In *ICSOC*, volume 7084 of *LNCS*, pp. 62–76. Springer, 2011.

[26] Y. Kabak and A. Dogac. A survey and analysis of electronic business document standards. *ACM Computing Surveys*, 42(3):11:1–11:31, 2010.

[27] D. Knuplesch, M. Reichert, W. Fdhila, and S. Rinderle-Ma. On enabling compliance of cross-organizational business processes. In *Proc. BPM*, volume 8094 of *LNCS*. Springer, 2013.

[28] J. Köpke and J. Eder. Logical invalidations of semantic annotations. In *Proc. CAiSE*, volume 7328 of *LNCS*, pp. 144–159. Springer, 2012.

[29] P. Langer, K. Wieland, M. Wimmer, and J. Cabot. EMF profiles: A lightweight extension approach for EMF models. *Journal of Object Technology*, 11(1), 2012.

[30] P. Leitner, J. Ferner, W. Hummer, and S. Dustdar. Data-driven and automated prediction of service level agreement violations in service compositions. *Distributed and Parallel Databases*, 31(3):447–470, 2013.

[31] L. T. Ly, D. Knuplesch, S. Rinderle-Ma, K. Göser, H. Pfeifer, M. Reichert, and P. Dadam. Seaflows toolset–compliance verification made easy for process-aware information systems. In *Proc. CAiSE Forum IST*, volume 72 of *LNBIP*, pp. 76–91. Springer, 2011.

[32] L. T. Ly, S. Rinderle-Ma, K. Göser, and P. Dadam. On enabling integrated process compliance with semantic constraints in process management systems. *Information Systems Frontiers*, 14(2):195–219, 2012.

[33] F. M. Maggi, M. Westergaard, M. Montali, and W. van der Aalst. Runtime verification of LTL-based declarative process models. In *Proc. RV*, volume 7186 of *LNCS*, pp. 131–146. Springer, 2012.

[34] P. Mayer and D. Lübke. Towards a BPEL unit testing framework. In *Workshop Proc. TAVWEB*, pp. 33–42. ACM, 2006.

[35] T. Mayerhofer, P. Langer, and M. Wimmer. xMOF: A semantics specification language for metamodeling. In , *Demos/Posters/StudentResearch@MoDELS*, volume 1115 of *CEUR Workshop Proceedings*, pp. 46–50. CEUR-WS.org, 2013.

[36] J. Mendling and M. Hafner. From WS-CDL choreography to BPEL process orchestration. *J. Enterprise Inf. Management*, 21(5):525–542, 2008.

[37] S. Mijatov, P. Langer, T. Mayerhofer, and G. Kappel. A framework for testing UML activities based on fUML. In *Workshop Proc. MoDeVVa*, volume 1069, pp. 1–10. CEUR-WS.org, 2013.

[38] D. Moran, L. Vaquero, and F. Galan. Elastically ruling the cloud: Specifying application's behavior in federated clouds. In *Proc. CLOUD*, pp. 89–96. IEEE, 2011.

[39] P.-A. Muller, F. Fleurey, and J.-M. Jézéquel. Weaving Executability into Object-Oriented Metalanguages. In *Proc. MoDELS*, volume 3713 of *LNCS*, pp. 264–278. Springer, 2005.

[40] C. Ouyang, M. Dumas, W. van der Aalst, A. H. T. Hofstede, and J. Mendling. From business process models to process-oriented software systems. *ACM transactions on software engineering and methodology*, 19(1):2, 2009.

[41] B. A. Schmit and S. Dustdar. Systematic design of web service transactions. In *Workshop Proc. TES*, LNCS, pp. 23–33. Springer, 2005.

[42] M. Strembeck and U. Zdun. An Approach for the Systematic Development of Domain-Specific Languages. *Software: Practice and Experience*, 39(15):1253–1292, 2009.

[43] S. Tai, P. Leitner, and S. Dustdar. Design by units: Abstractions for human and compute resources for elastic systems. *IEEE Internet Computing*, 16(4):84–88, 2012.

[44] H. Tran, T. Holmes, U. Zdun, and S. Dustdar. *Handbook of Research on Business Process Modeling*, chapter Modeling Process-Driven SOAs, pp. 27–48. IGI Global Hershey, 2009.

[45] H. Tran, U. Zdun, and S. Dustdar. VbTrace: using view-based and model-driven development to support traceability in process-driven SOAs. *Software and System Modeling*, 10(1):5–29, 2011.

[46] A. Vallecillo. On the combination of domain specific modeling languages. In *Proc. ECMFA*, volume 6138 of *LNCS*, pp. 305–320. Springer, 2010.

[47] W. van der Aalst. Business process management: A comprehensive survey. *ISRN Software Engineering*, 2013, 2013.

[48] W. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf. Multiparty contracts: Agreeing and implementing interorganizational processes. *The Computer Journal*, 53(1):90–106, 2010.

[49] W. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23(2):99–113, 2009.

[50] W. van der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.

[51] W. van der Aalst and M. Weske. The P2P approach to interorganizational workflows. In *Proc. CAiSE*, volume 2068 of *LNCS*, pp. 140–156. Springer, 2001.

[52] M. Vujasinovic, N. Ivezic, B. Kulvatunyou, E. Barkmeyer, M. Missikoff, F. Taglino, Z. Marjanovic, and I. Miletic. Semantic mediation for standard-based B2B interoperability. *IEEE Internet Computing*, 14(1):52–63, 2010.

[53] 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.

[54] Y. Wei and M. Blake. Decentralized resource coordination across service workflows in a cloud environment. In *Workshop Proc. WETICE*, pp. 15–20. IEEE, 2013.

[55] M. Weidlich, A. Polyvyanyy, J. Mendling, and M. Weske. Causal behavioural profiles–efficient computation, applications, and evaluation. *Fundamenta Informaticae*, 113(3):399–435, 2011.

[56] M. T. Wynn, H. Verbeek, W. van der Aalst, A. H. ter Hofstede, and D. Edmond. Business process verification–finally a reality! *Business Process Management Journal*, 15(1):74–92, 2009.

[57] J. Yan, Z. Li, Y. Yuan, W. Sun, and J. Zhang. BPEL4WS unit testing: Test case generation using a concurrent path analysis approach. In *Proc. ISSRE*, pp. 75–84. IEEE, 2006.

[58] S. Zschaler, D. Kolovos, N. Drivalos, R. Paige, and A. Rashid. Domain-specific metamodelling languages for software language engineering. In *Proc. SLE*, volume 5969 of *LNCS*, pp. 334–353. Springer, 2010.

[59] S. Zugal, J. Pinggera, and B. Weber. Creating declarative process models using test driven modeling suite. In *CAiSE Forum IS Olympics*, volume 107 of *LNBIP*, pp. 16–32. Springer, 2012.