A Survey of Formal Business Process Verification:
From Soundness to Variability

Heerko Groefsema and Doina Bucur
Johann Bernoulli Institute for Mathematics and Computer Science,
University of Groningen, Nijenborgh 9, 9747 AG Groningen, the Netherlands
$\{$*h.groefsema, d.bucur*$\}$*@rug.nl*

Keywords: Business Process Management, Verification, Model Checking, Survey.

Abstract: Formal verification of business process models is of interest to a number of application areas, including checking for basic process correctness, business compliance, and process variability. A large amount of work on these topics exist, while a comprehensive overview of the field and its directions is lacking. We provide an overview and critical reflections on existing approaches.

1 INTRODUCTION

Business process modeling helps businesses to increase the quality of their processes. Formal techniques are used to model, implement, execute, and monitor business process models. *Model checking* is a technique which verifies a given system model for compliance with a specification of interest, for various practical *goals* including ensuring basic correctness of processes, business compliance checking, and process variability. A related survey (Morimoto, 2008) provides an overview of business process checking, but does not consider compliance and variability as supported through formal verification, while in (Aiello et al., 2010), we survey variability for business processes.

While a large amount of work exists in the field of business process verification, it lacks an overview of the state of the field and its related formal verification frameworks. As such, in the present treatment, we aim to provide an overview of formal verification goals, techniques, and frameworks for business process modeling, and give critical reflections.

Frameworks aiming at the verification of business process models exist. They supports various process-specification formalisms, e.g., *imperative*, *declarative*, *event-driven*, or *artifact-centric*. In *imperative* specification formalisms, processes are modelled as sets of *tasks* or *activities*, *gates*, and *events* interlinked by *flows* or *transitions*. Each activity describes a single unit of work and the transitions describe the order between these units of work. Common notations include Business Process Model and Notation (BPMN) (OMG, 2011), Business Process Execution Language (BPEL) (Oasis, 2007), Unified Modeling Language (UML) activity diagrams (OMG, 2011), and Yet Another Workflow Language (YAWL) (Hofstede et al., 2010). Alternatively, *declarative* specification formalisms, such as (Pesic and van der Aalst, 2006), model processes without distinct flow controls which specify order between units of work. Instead, these specifications express a process model as a set of activities and a set of *constraints* over these activities, with the constraints restricting the possible inclusion and ordering of the activities. Any process behaviour not prohibited by these constraints is valid.

Event-driven specification is another approach to business process modelling. Defined in (Keller et al., 1992), Event-driven Process Chains (EPC) are directed graphs mainly consisting of events, functions (activities), and logical connectors (gates). Unlike imperative specifications, EPC do not model node ordering explicitly. Although EPC are known for their intelligible notation and simplicity, their lack of semantics is a topic of discussion (van der Aalst, 1999). Finally, *artifact-centric* specifications focus on the evolution of business entities and data. Originally proposed in (Nigam and Caswell, 2003), such specifications incorporate the notion of the lifecycle of business artifacts, such as data—which is ignored by most other specifications.

In what follows, Section 2 examines the techniques and goals of business processes verification. We classify and discuss verification frameworks in Section 3, and conclude in Section 4.

2 BUSINESS PROCESS MODEL CHECKING

Business processes verification can be based on several algorithmic techniques (and some supporting intermediary modelling formalisms), and be used for a number of goals. Next we overview these.

2.1 VERIFICATION TECHNIQUES

Petri Nets are state-transition systems used to analyze distributed systems. They are amenable to intuitive graphical notation which, unlike most of the process notations discussed in Section 1, include a mathematical definition of its execution semantics. A number of subclasses of Petri Nets have been defined, most notably the sub-class of Workflow (WF) nets (van der Aalst, 1998) which is applied as an intermediary formalism in the verification of business processes. Besides Petri Nets, a system may also be modelled as a *finite state machine* (FSM)—a directed graph of nodes and edges, with nodes representing a system state system and edges representing a change in state. Both Petri Nets and FSM-like models are then verified against a *specification* or *correctness property*.

On the other hand, generic model checkers, e.g., SPIN (Holzmann, 2004) and nuSMV2 (Cimatti et al., 1999), implement search algorithms to verify any system modelled in the model checker's *input languages*. Of these, notable is the Process Meta-Language, or Promela, used by SPIN. Business processes can be remodelled using Promela, with the Promela implementation then internally translated into an automaton and verified against a correctness property.

Correctness properties may be informal specifications, process models, or logic formulas. Informal specifications include properties defined as simple tuples or programming methods. Process models themselves can be used as correctness properties as well. In this case, the original business process model is verified to be a refinement of the correctness model. Logic properties are formulas in logics ranging from propositional logic to *deontic* and *temporal logic*. Deontic logics reason about obligations and permissions. Temporal logics include Linear Temporal Logic (LTL) and Computation Tree Logic (CTL), a branching-time logic. LTL specifies properties (e.g., the universality of a certain state property, and the order of states) over states occurring on process execution paths. CTL extends this set of temporal operators with path quantifiers, such that formulas can specify properties over branching executions. Extensions of these logics are common for process verification. They include Past-time LTL (PLTL), but also novel logics for business process specification.

2.2 GOALS OF VERIFICATION

Business process verification is the act of determining if a business process model is correct with regard to a set of formal correctness properties. Often, verification is automated by tools known as analyzers or model checkers. Several goals for using verification are presented in the business process literature.

The first goal is verifying basic properties such as *reachability* and *termination*. Reachability of a business activity requires an execution path to exist leading to that activity starting from the initial activities. A termination property requires that all possible execution traces terminate. Business process *soundness*, a property originally proposed in the area of Petri Net verification, is known as the combination of these two properties plus a third: the absence of related running activities at process termination (i.e., proper completion). Avoiding the deployment of erroneous processes that do not conform with these properties is obviously advantageous: *"[erroneously] designed workflow models can result in failed workflow processes, execution errors, and disgruntled customers and employees"* (Bi and Zhao, 2004).

The second goal for business process verification is business *compliance*. Compliance checks whether process models conform with specifications, which in this case can be another process model or a set of rules, such as (inter)national laws and standards. When verifying compliance, rules are often specified using a formal logic over the entities (e.g., events, activities) of the business process model. In other cases, these rules are informally specified. For example, regulations could specify that before processing a wire transfer, a bank should identify if any sanctions exist regarding the involved parties.

The third goal of verification of process models, *variability*, extends upon compliance. *"In the context of BPM, variability indicates that parts of a business process remain variable, or not fully defined, in order to support different versions of the same process depending on the intended use or execution context"* (Aiello et al., 2010). Variability aims to support different versions of the same process. This includes support of process families at design-time, when a new process *variant* can be derived from a generic process, and process flexibility or adaptability at runtime, where a generic process can be adapted. Variability can be specified in two different ways. The first, which is not in the scope of this survey, employs the use of variation points to provide different options at specific points in a process. The second, which is in scope, uses rules like those of compliance to specify how each version of a process should behave.

A final goal of business process verification

Table 1: Soundness tools and frameworks.

Framework	Formalisms			Tool
	Modelling	Intermediate	Correctness properties	
(van der Aalst, 1998)	Petri Net			Woflan
(Bi and Zhao, 2004)	WfMC		Propositional logic	Algorithmic
(Choi and Zhao, 2005)	WfMS			
(van Dongen et al., 2007)	EPC	Petri Net		ProM
(Fisteus et al., 2005)	BPEL4WS	CFM	LTL, CTL	SPIN, SMV
(Karamanolis et al., 2000)	WfMS	FSP		LTSA
(Koehler et al., 2002)	Imperative	FSM	CTL	nuSMV
(Masalagiu et al., 2009)	BPMN	Petri Net to TLA+	LTL, CTL (via TLA+)	TLC
(Nakajima, 2006)	BPEL	EFA to Promela	LTL	SPIN
(Weber et al., 2010)	Annotated process		Annotated process	Algorithmic
(Wynn et al., 2009)	YAWL	Petri Net		YAWL

(which we do not cover in this survey) deals with processes including multiple parties, such as business process collaborations (De Backer et al., 2009). The goal of verification includes, for example, the *compatibility* between processes, or lanes.

3 OVERVIEW OF FRAMEWORKS

The existing frameworks aim to verify business processes either at design time or at runtime. The most basic form of verification of processes deals with design-time verification of soundness properties, e.g., termination and reachability. Table 1 gives an overview of these frameworks. (van der Aalst, 1998) introduced soundness to the field of BPM by translating workflows into Petri Nets, and (Wynn et al., 2009) perfected the application by allowing Or-joins and cancelation regions. Due to this, Petri Nets are commonly used as intermediate formalisms by soundness verification frameworks, including (van Dongen et al., 2007), who use it to verify EPC. Another popular method is by translating processes into a model checker input language, e.g.: (Masalagiu et al., 2009) verify BPMN by translating it (via a Petri Net intermediate model) into the model checker input language TLA+, (Karamanolis et al., 2000) translate processes to the process algebra FSP and checks the result with the Labeled Transition System Analyzer, (Koehler et al., 2002) translate into the nuSMV input language, and (Nakajima, 2006) translates into the SPIN language Promela.

Other frameworks verify business compliance; an overview of these is given in Table 2. A dominant number of compliance frameworks focus on verifying imperative specifications such as BPMN, BPEL, EPC, and UML sequence diagrams. (Anderson et al., 2005; Arbab et al., 2009; Awad et al., 2008; Foster et al., 2003; Ghose and Koliadis, 2007; Goedertier and Vanthienen, 2006; Janssen et al., 1998; Liu et al., 2007; Ly et al., 2008; Ly et al., 2011; Nakajima, 2002) all belong in this category. Others extend declarative specifications with compliance features. In (Chesani et al., 2009), compliance is modelled based on DecSerFlow, a declarative runtime process specification, by translating it into a reactive event calculus. (Pulvermueller et al., 2010) aims at verifying the compliance of design-time EPC using an extension of CTL that differentiates between events and functions. Finally, (Deutsch et al., 2009) proposes verifying the compliance of artifact-centric processes against properties expressed in an extension of LTL.

The last set of frameworks (listed in Table 3) aim at supporting variability. Declarative variability extends upon compliance by only specifying rules over the set of tasks in a process, instead of building an imperative graph. As such, the authors of (Governatori et al., 2006) first propose a compliance framework based upon the newly proposed deontic logic FCL, then continue by extending this framework, in (Governatori et al., 2011), with goals to provide a fully declarative description. (Sadiq et al., 2005) proposes defining pockets of flexibility within imperatively specified processes to introduce design-time variability, using constraints which provide ordering and inclusion information but which are not specified using a formal logic. Formal frameworks base their declarative specifications on the temporal logics LTL and CTL. As examples, (Demeyer et al., 2010) use Finite LTL to specify fully declarative processes, (D'Aprile et al., 2011) specify declarative processes using temporal Answer Set Programming (ASP) and Dynamic LTL, (Pesic and van der Aalst, 2006) use LTL to specify flexible run-time processes, and the related work of (van der Aalst and Pesic, 2006) aims towards service flows instead. Finally, (Maggi et al., 2011) extends upon the work of (Pesic and van der Aalst, 2006) to provide for runtime recovery after breaking constraint compliance. (Groefsema et al.,

Table 2: Compliance tools and frameworks.

Framework	Formalisms: Modelling	Formalisms: Intermediate	Formalisms: Correctness properties	Tool
(Anderson et al., 2005)	UML Sequence Diagram	CSP	CSP	FDR
(Arbab et al., 2009)	BPMN	Reo	Automata	Vereofy/ mCRL2
(Awad et al., 2008)	BPMN	Petri Net	PLTL	LoLA/ nuSMV
(Chesani et al., 2009)	DecSerFlow		Event Calc.	Algorithmic
(Deutsch et al., 2009)	Business artifacts		LTL-FO	Algorithmic
(Foster et al., 2003)	UML+ BPEL	FSP		LTSA
(Gerede and Su, 2007)	Business artifacts		CTL-FO	Algorithmic
(Ghose and Koliadis, 2007)	Annotated BPMN	Annotated digraphs	Process effect rules	Algorithmic
(Goedertier and Vanthienen, 2006)	BPMN		PENELOPE	Prolog CLP(fd)
(Janssen et al., 1998)	AMBER	Promela	LTL	SPIN
(Liu et al., 2007)	BPEL	Pi-calculus to FSM	LTL	nuSMV
(Ly et al., 2008; Ly et al., 2011)			CRG	
(Montali et al., 2010)	LTL	ALP	ALP	SCIFF
(Nakajima, 2002)	WSFL	Promela	LTL	SPIN
(Pulvermueller et al., 2010)	EPC		EG-CTL	BAM
(Weber et al., 2008)	Annotated Process Graph			

2011) uses CTL to define process templates; processes based upon such a template are then verified for compliance with that template at design-time. Finally, (Bulanov et al., 2011) proposes Temporal Process Logic (TPL) to provide a formal mechanism supporting different gates to merge processes.

Critically, drawbacks exist in certain frameworks for compliance and variability. Some frameworks inefficiently translate the business-process model into a model checker input language, introducing a large *overhead* in the ensuing state space (e.g., a simple process of five activities and four transitions is reportedly mapped to 201 states and 586 transitions in SPIN by (Nakajima, 2002)).

Other methods for design-time verification (but not for runtime verification, which checks linear execution paths) lack good support for complex branching features of the modelling formalism (e.g., do not support parallel gates, or execution loops). To improve on this, some introduce workarounds, e.g., (Pulvermueller et al., 2010) proposes using simple variables on automata to fork exclusive paths and synchronize parallel paths. In other work (Feja et al., 2009), the same authors propose an unsound Kripke translation when dealing with parallel paths. Here, pairs of activities from different parallel paths are explicitly synchronized; in reality, however, each path should only be synchronized after a join. Other frameworks, e.g., (Sadiq et al., 2005), (Choi and Zhao, 2005), and (Groefsema et al., 2011), simply ignore exclusive, inclusive, and/or parallel paths for reasons of complexity. (Weber et al., 2008) proposes two verification algorithms in polynomial time; however, one is reported by the authors as unsound and complete, the other sound but incomplete, and both only support acyclic processes. (Montali et al., 2010) reports its verification algorithm to be unable to terminate under certain loops.

Finally, other frameworks require users to apply newly proposed or extended specification logics in order to specify correctness properties. Examples include (Pulvermueller et al., 2010), which extends CTL with the ability to differentiate between events and functions in EPC, (Governatori et al., 2006), which proposes an entirely new deontic logic, and (Bulanov et al., 2011), which proposes a new temporal process logic to allow process mergers.

4 CONCLUSIONS

Formal verification of business processes was initially proposed to check for process soundness, but lately has been deployed to also support business compliance and variability. While process soundness is well supported by frameworks based on Petri-Net formalisms, the areas of compliance and variability checking lack design-time solutions which (i) minimize the overhead of states in the formal model, and (ii) support large subsets of the business process modelling formalism, including parallel gates and process loops.

Table 3: Variability tools and frameworks.

Framework	Formalisms			Tool
	Modelling	Intermediate	Correctness properties	
(D'Aprile et al., 2011)	Temporal ASP		Temporal ASP	
(Bulanov et al., 2011)	Imperative		TPL	
(Governatori et al., 2011)	BPMN		FCL	
(Groefsema et al., 2011)	BPMN+ CTL		CTL	VxBPMN
(Demeyer et al., 2010)	Saturn	Automata	Finite LTL	Saturn Eng.
(Pesic and van der Aalst, 2006; van der Aalst and Pesic, 2006; Maggi et al., 2011)	LTL	Automata	LTL	Declare
(Rychkova et al., 2008)	BPMN + FO	Alloys	FO	Alloys
(Sadiq et al., 2005)			Informal	Chameleon

REFERENCES

Aiello, M., Bulanov, P., and Groefsema, H. (2010). Requirements and tools for variability management. In *Proc. IEEE 34th Annual Computer Software and Applications Conference Workshops*, COMPSACW '10, pages 245–250, Washington, DC, USA.

Anderson, B., Hansen, J. V., Lowry, P., and Summers, S. (2005). Model checking for E-business control and assurance. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35(3):445–450.

Arbab, F., Kokash, N., and Meng, S. (2009). Towards using Reo for compliance-aware business process modeling. In Margaria, T. and Steffen, B., editors, *Leveraging Applications of Formal Methods, Verification and Validation*, volume 17 of *Communications in Computer and Information Science*, pages 108–123. Springer.

Awad, A., Decker, G., and Weske, M. (2008). Efficient compliance checking using BPMN-Q and temporal logic. In *Proc. 6th Int. Conf. on Business Process Management*, BPM '08, pages 326–341. Springer.

Bi, H. H. and Zhao, J. L. (2004). Applying propositional logic to workflow verification. *Information Technology and Management*, 5:293–318.

Bulanov, P., Lazovik, A., and Aiello, M. (2011). Business process customization using process merging techniques. In *Int. Conference on Service-Oriented Computing and Applications (SOCA 2011)*, pages 1–4.

Chesani, F., Mello, P., Montali, M., and Torroni, P. (2009). Web services and formal methods. chapter Verification of Choreographies During Execution Using the Reactive Event Calculus, pages 55–72. Springer.

Choi, Y. and Zhao, J. L. (2005). Decomposition-based verification of cyclic workflows. In *Proc. Third Int. Conf. on Automated Technology for Verification and Analysis*, ATVA'05, pages 84–98. Springer.

Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M. (1999). NUSMV: a new Symbolic Model Verifier. In N. Halbwachs and D. Peled, editors, *Proc. 11th Conference on Computer-Aided Verification (CAV)*, volume 1633 of *LNCS*, pages 495–499. Springer.

D'Aprile, D., Giordano, L., Gliozzi, V., Martelli, A., Pozzato, G. L., and Dupré, D. T. (2011). Verifying compliance of business processes with temporal answer sets. In *CILC*, pages 147–161.

De Backer, M., Snoeck, M., Monsieur, G., Lemahieu, W., and Dedene, G. (2009). A scenario-based verification technique to assess the compatibility of collaborative business processes. *Data Knowl. Eng.*, 68(6):531–551.

Demeyer, R., Van Assche, M., Langevine, L., and Vanhoof, W. (2010). Declarative workflows to efficiently manage flexible and advanced business processes. In *Proc. 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming*, PPDP, pages 209–218. ACM.

Deutsch, A., Hull, R., Patrizi, F., and Vianu, V. (2009). Automatic verification of data-centric business processes. In *Proc. 12th Int. Conf. on Database Theory*, ICDT, pages 252–267. ACM.

Feja, S., Speck, A., and Pulvermüller, E. (2009). Business process verification. In *GI Jahrestagung*, pages 4037–4051.

Fisteus, J. A., Fernández, L. S., and Kloos, C. D. (2005). Applying model checking to bpel4ws business collaborations. In *Proc. ACM Symposium on Applied Computing (SAC)*, pages 826–830. ACM.

Foster, H., Uchitel, S., Magee, J., and Kramer, J. (2003). Model-based verification of web service compositions. In *Proc. 18th IEEE Int. Conf. on Automated Software Engineering (ASE)*, pages 152–163.

Gerede, C. and Su, J. (2007). Specification and verification of artifact behaviors in business process models. In Krämer, B. J., Lin, K.-J., and Narasimhan, P., editors, *Service-Oriented Computing (ICSOC)*, volume 4749 of *LNCS*, pages 181–192. Springer.

Ghose, A. and Koliadis, G. (2007). Auditing business process compliance. In *Proc. 5th Int. Conf. on Service-Oriented Computing*, ICSOC, pages 169–180. Springer.

Goedertier, S. and Vanthienen, J. (2006). Designing compliant business processes with obligations and permissions. In *Proc. Int. Conf. on Business Process Management Workshops*, BPM, pages 5–14. Springer.

Governatori, G., Milosevic, Z., and Sadiq, S. (2006). Compliance checking between business processes and business contracts. In *Proc. 10th IEEE Int. Conf. En-*

terprise Distributed Object Computing, EDOC, pages 221–232. IEEE Computer Society.

Governatori, G., Olivieri, F., Scannapieco, S., and Cristani, M. (2011). Designing for compliance: norms and goals. In *Proc. 5th Int. Conf. on Rule-based modeling and computing on the semantic web*, RuleML, pages 282–297. Springer.

Groefsema, H., Bulanov, P., and Aiello, M. (2011). Declarative enhancement framework for business processes. In *Int. Conference on Service-Oriented Computing, ICSOC, LNCS 7084*, pages pp 495–504.

Hofstede, A. H., Russell, N., Aalst, W. M., and Adams, M. (2010). *Modern Business Process Automation: YAWL and its Support Environment*. Springer.

Holzmann, G. (2004). The SPIN Model Checker. A Primer and Reference Manual.

Janssen, W., Mateescu, R., Mauw, S., and Springintveld, J. (1998). Verifying business processes using SPIN. In *Proc. of the 4th Int. SPIN Workshop*, pages 21–36.

Karamanolis, C. T., Giannakopoulou, D., Magee, J., and Wheater, S. M. (2000). Model checking of workflow schemas. In *Proc. 4th Int. Conf. on Enterprise Distributed Object Computing*, EDOC, pages 170–181. IEEE Computer Society.

Keller, G., Nüttgens, M., and Scheer, A. W. (1992). Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Technical Report 89, Universität des Saarlandes, Germany.

Koehler, J., Tirenni, G., and Kumaran, S. (2002). From business process model to consistent implementation: a case for formal verification methods. In *Proc. 6th Int. Conf. on Enterprise Distributed Object Computing (EDOC)*, pages 96–106.

Liu, Y., Müller, S., and Xu, K. (2007). A static compliance-checking framework for business process models. *IBM Syst. J.*, 46(2):335–361.

Ly, L. T., Rinderle, S., and Dadam, P. (2008). Integration and verification of semantic constraints in adaptive process management systems. *Data Knowl. Eng.*, 64(1):3–23.

Ly, L. T., Rinderle-Ma, S., Knuplesch, D., and Dadam, P. (2011). Monitoring business process compliance using compliance rule graphs. In *Proc. Confederated Int. Conf. On the move to meaningful internet systems - Volume I*, OTM, pages 82–99. Springer-Verlag.

Maggi, F., Montali, M., Westergaard, M., and Aalst, W. (2011). Monitoring business constraints with linear temporal logic: An approach based on colored automata. In *Business Process Management*, volume 6896 of *LNCS*, pages 132–147. Springer.

Masalagiu, C., Chin, W.-N., Andrei, S., and Alaiba, V. (2009). A rigorous methodology for specification and verification of business processes. *Formal Aspects of Computing*, 21:495–510.

Montali, M., Torroni, P., Chesani, F., Mello, P., Alberti, M., and Lamma, E. (2010). Abductive logic programming as an effective technology for the static verification of declarative business processes. *Fundam. Inf.*, 102(3-4):325–361.

Morimoto, S. (2008). A survey of formal verification for business process modeling. In *Proc. 8th Int. Conf. on Computational Science, Part II*, ICCS, pages 514–522. Springer-Verlag.

Nakajima, S. (2002). Verification of Web service flows with model-checking techniques. In *Proc. First Int. Symposium on Cyber Worlds*, pages 378–385.

Nakajima, S. (2006). Model-checking behavioral specification of BPEL applications. *Electron. Notes Theor. Comput. Sci.*, 151(2):89–105.

Nigam, A. and Caswell, N. S. (2003). Business artifacts: An approach to operational specification. *IBM Syst. J.*, 42(3):428–445.

Oasis (2007). Web Services Business Process Execution Language Version 2.0. Technical report, Oasis.

OMG (2011). Business Process Model And Notation (BPMN) Version 2.0. Technical Report 2011-01-03.

OMG (2011). OMG Unified Modeling Language^{TM} (OMG UML). Technical Report 2011-08-05.

Pesic, M. and van der Aalst, W. M. P. (2006). A declarative approach for flexible business processes management. In *Proc. Int. Conf. on Business Process Management Workshops*, BPM, pages 169–180. Springer-Verlag.

Pulvermueller, E., Feja, S., and Speck, A. (2010). Developer-friendly verification of process-based systems. *Knowl.-Based Syst.*, 23(7):667–676.

Rychkova, I., Regev, G., and Wegmann, A. (2008). Using declarative specifications in business process design. *Int. J. of Computer Science and Applications*, 5(3b):45–68.

Sadiq, S. W., Orlowska, M. E., and Sadiq, W. (2005). Specification and validation of process constraints for flexible workflows. *Inf. Syst.*, 30(5):349–378.

van der Aalst, W. M. (1999). Formalization and verification of event-driven process chains. *Information and Software technology*, 41(10):639–650.

van der Aalst, W. M. P. (1998). The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66.

van der Aalst, W. M. P. and Pesic, M. (2006). DecSerFlow: Towards a truly declarative service flow language. *LNCS: Web Services and Formal Methods, Volume 4184*, pages 1–23.

van Dongen, B. F., Jansen-Vullers, M. H., Verbeek, H. M. W., and van der Aalst, W. M. P. (2007). Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. *Comput. Ind.*, 58(6):578–601.

Weber, I., Governatori, G., and Hoffmann, J. (2008). Approximate compliance checking for annotated process models. In *Advances in Enterprise EngineeringProceedings of the GRCIS workshop at CAiSE*, volume 8.

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

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