Towards a Compiler for Business Processes — A Research Agenda

Thomas M. Prinz, Thomas S. Heinze,
and Wolfram Amme

Chair of Software Technology, Friedrich Schiller University
Jena, Germany
Email: {Thomas.Prinz, T.Heinze,
Wolfram.Amme}@uni-jena.de

Johannes Kretzschmar
and Clemens Beckstein

Artificial Intelligence Group, Friedrich Schiller University
Jena, Germany
Email: {Johannes.Kretzschmar,
Clemens.Beckstein}@uni-jena.de

***Abstract*—Business process management (BPM) and service-oriented architectures (SOA) promise the development, application, maintenance, and improvement of business processes, i.e., service compositions, as it is done in software engineering. However, BPM is currently more similar to an unfinished patchwork and an overall system supporting BPM is missing since it requires a unified execution engine (a virtual machine), a common intermediate representation, and eventually a compiler. In this paper, we motivate the construction of such a system for BPM and propose an approach including the mentioned sub systems. Additionally, we show the gaps in current approaches and why some techniques are not yet fully applicable. We encourage that system with state-of-the-art approaches and our own ideas of BPM, compiler construction, and artificial intelligence. Such a system finally will encourage processes for small and medium-sized enterprises and for SOA applications.**

Keywords–Business Process Management; Compiler; Intermediate Representation; Planning; Service-oriented Architecture.

I. INTRODUCTION

It seems that time has come for the efficient and successful application of business process management (BPM) and service-oriented architectures (SOA). There are promising approaches and techniques for each step of the BPM life cycle [1], i.e., (1) requirements analysis, (2) design, (3) implementation, (4) verification and testing, and (5) ongoing improvements. However, Koehler et al. have already emphasized gaps in the BPM life cycle (especially the missing automation of process translations into executable processes) which hinder companies in exploiting the benefits of BPM [2]. Therefore, BPM is more similar to an unfinished patchwork and a unified system seriously supporting BPM is needed.

In this paper, we argue for a compiler for business processes, i.e., service compositions. Koehler et al. have already argued for a compiler for business IT-systems and provide interesting approaches and ideas. Their compiler follows a top-down approach. However, the implementation of a compiler for business processes needs both: A well-defined business process modeling language as input and a machine that executes a process for the output. For example, the programming language Java would not have been so successful if it had not used its own virtual machine abstracting from the real physical design.

A virtual machine for business processes is similar to a unified and sufficient intermediate representation (IR) (like a bytecode for processes). Current process description languages like the Business Process Model and Notation 2.0 (BPMN) [3] and Event-driven Process Chains (EPCs) [4] are primarily designed for high level descriptions of business processes rather than for technical implementations. Although we do *not* want to translate abstract processes into executable ones (since there is the need for developers supporting that transformation), a well-defined and common technical basis for the definition of usable constructs and expressions is needed to guarantee such a transformation without later risking high additional effort.

As the IR is *not* suitable for the development of processes in general (like machine code or Java Bytecode for programs), it is necessary to provide a more high-level but IR-conform processing language (like a subset of BPMN and EPCs). Therefore, that language has to be automatically transformable into the IR. Such a transformation can basically be done by a compiler. The compilation process should provide powerful tools and analyses with useful failure and diagnostic information about the process for the developer. These kinds of information are currently undetailed and so new algorithms have to fill the gap. Additionally, a tool can handle such information to provide several options for (semi-) automatic error handling. Within a dialog between the developer and the system, the developer can choose the best fitting solution.

Although there are already approaches considering parts of the mentioned system, little attention has been paid to their interaction. We identified four important subsystems for a general overall BPM system: *(1)* a simple and unified process engine, i.e., a (virtual) machine, *(2)* a unified IR, *(3)* a verifying compiler translating business processes into that IR, and *(4)* an error handling which provides correction proposals being applied to processes.

In this paper, we consider the state-of-the-art of BPM systems (Section II) in short. We describe a system consisting of a compiler, an IR, and a process engine with regard to state-of-the-art approaches, solutions (Section III) and provide own research approaches to finally enable the implementation of such a new approach for a system. Section IV summarizes and concludes the paper.

II. STATE OF THE ART

There are many tools providing BPMN for BPM e.g., Activiti BPM Platform [5], Redhat jBPM [6], IBM WebSphere [7], AristaFlow BPM Suite [8], and BonitaBPM [9]. These tools allow for the development, simulation, and execution of business processes. Furthermore, they have additional features supporting parts of the BPM lifecycle.

 ISBN: 978-1-61208-387-2

However, most of those tools use different subsets of and (intermediate) representations for BPMN such that processes are not interchangeable between tools without additional effort. Considering that fact for the programming language Java for example: It would be strange if there would be a variety of virtual machines for Java, for each accepting a different subset of Java bytecode instructions. *Apromore* [10] is an advanced process model repository based on a common intermediate representation (canonical representation) to handle different process model languages. However, although the repository benefits from that representation, common parts of process modeling languages like exceptions, exception handling, signals, transitions, etc. are not accurately representable.

Our approach calls for the implementation of a core virtual machine for business process modeling languages that is extensible by additional tools. Such a core virtual machine asks for a greatest possible subset of process elements being accepted by the machine (the IR) and also asks for a compiler, which constructs the IR form for current process modeling languages (e.g., *Web Services Business Process Execution Language 2.0* (BPEL) [11], *Yet Another Workflow Language* (YAWL) [12], EPCs or BPMN). The compilation process is not always straightforward as some modeling languages (e.g., BPMN) only provide subsets of executable process models (e.g., BPMN Process Execution Conformance [3] for BPMN).

Compilers for business processes are rare in existing tools. Most of them take a process as it is and interpret it stepwise. Sometimes, however, additional information is needed for the execution of a process which can be derived from a compiler, e.g., data types, soundness or reference safety. Additionally, most business process modeling language's output formats are not suitable for fast and efficient analyses and compilers therefore have to create a more compact format.

For this purpose, our approach calls for the implementation of a compiler that transforms and analyzes a process in an intermediate and interchangeable representation. The most common used intermediate representations for business processes are specification conform exchange formats or BPEL. BPEL has the great advantage to be a block-based language. That, however, is the largest problem for simply transforming processes of graph-based languages (like BPMN) to BPEL [13][14]. Furthermore, BPEL was designed to orchestrate different web services and not to directly execute tasks.

Our approach relies on an intermediate representation for business processes. That representation should allow and outperform existing analyses for the verification of pre-defined process properties since process validation is very expensive by currently supported BPM tools. State-of-the-art research considers those verification mechanisms. For example, in previous work [15], we have focused on compiler-based mechanisms for finding deadlocks and missing synchronizations. These techniques are so efficient that we were able to perform the analyses after each modification of a process model and to give detailed diagnostic information as shown by our tool implementation [16].

Besides these compiler-based mechanisms, we argue for semantic analyses of processes by artificial intelligence (AI) planning methods. These methods rely on semantic descriptions of process-activities. Semantic descriptions are already widely used in the field of service-oriented architectures through service description standards, like *Web Service Modeling Language* (WSML) [17] and *Web Ontology Language for Semantic Web Services* (OWL-S) [18]. In contrast to previous service description standards, these languages allow the specification of requirements and impacts of a service regarding a descriptive domain model. With such descriptions, an AI planner is able to goal-oriented generate an ordered set of services, which can be executed as a BPEL-like service composition by workflow engines [19][20]. Because of the fast growing complexity of planning problems, there are usually assumptions of the domain models concerning time, execution, observability and influence aspects [21]. Therefore, AI planning for workflow generation is only practical in particular use-cases with simple workflow models. Our approach focuses on cheaper methods for evaluating a process, which are part of planning algorithms. Thereby, AI planning could be used also for more comprehensive workflow descriptions and could enhance a comprehensive semantic analysis.

III. Compiler-Engine Architecture for Business Processes

In the following, we propose a new BPM architecture. For this purpose, we explain the overall system first and subsequently describe each subsystem in detail.

Figure 1 gives a structural overview of the complete system. The system has two sides inspired by Amme et al. [22]: a *producer* and a *consumer side*. The producer side is a compiler being adaptable to each business process development tool. It accepts an entire process in different process modeling languages, where a specific front-end containing a parser and a transformation exists for each language. The internal format is an IR, which allows for the application of semantic analyses whose output in turn can be used as input for an error handler, a coder, and an annotator.

The consumer side consists of an engine and virtual machine, respectively. It loads a compiled process and extracts the IR. Then, it executes the IR and performs dynamic semantic analyses and, in the case of an error, it provides an error handling which enables for a "rescue" of the running process.

The interface between the producer and consumer side is a *business process repository*. The producer side stores compiled processes within that repository whereas the consumer side can load them. Furthermore, error handling systems on both sides utilize the same repository for their analyses.

A. Producer Side

The producer side is divided into a *parser*, a *transformation*, an *IR*, *semantic analyses*, and an *error handling* as well as a *coder* and an *annotator*.

1) Parsing and Transformation: The parser's task is to structurally analyze and verify the entire process against its description language, i.e., a conformance check. Afterwards, the transformer translates that process into an IR. For this purpose, the front-end, consisting of the parser and the transformer, depends on the process language.

Two approaches can be distinguished: (1) Defining a mapping from one language to the other [23][24][25] or (2) creating a parse tree (process structure tree, PST) which is then used for a translation [26]. Both approaches have their roots in compiler theory. However, since the PST is similar

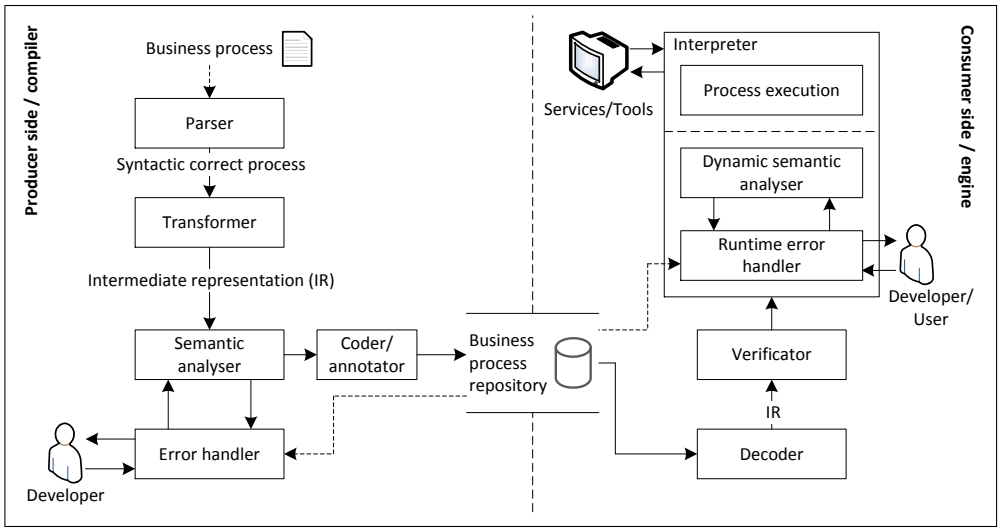

Figure 1. System overview containing a compiler and an engine.

to an abstract syntax tree (AST), it provides more structural information and therefore we prefer the PST as a mapping is still possible at a later time.

The inclusion of instructions and variables of the business process constitutes the major problem during the parsing and transformation. Process instructions can be translated as shown by Amme et al. [27], using the Concurrent Static Single Assignment Form (CSSA form) [28]. However, the creation of CSSA form is currently only suitable for structured graphs. Most business processes are unstructured, so we have to define a transformation for those processes and their instructions.

2) Intermediate Representation: A common IR must cover (almost) all constructs and instructions of currently popular process languages. Furthermore, it must provide and support efficient techniques for detailed semantic analyses. Currently, Petri nets [29] and workflow graphs [30] are commonly used to represent language-independent and analyzable processes. Since workflow graphs have Petri net semantics but provide more structural information, workflow graphs should be used as they are very similar to (concurrent) control flow graphs of compiler theory [28].

In previous work, we have defined an *extended workflow graph* (eWFG) based on CSSA form [27][31]. In the next steps, we plan to extend those eWFGs with advanced language constructs like OR-joins, events, signals, transactions, exception handling, and roles.

3) Semantic Analyses: The task of the semantic analyser is to verify the IR against properties and to restructure the IR, e.g., for the encoding into a mobile format. Semantic analyses consist of structural, context-sensitive, content-related, and goal-oriented analyses. Structural analyses consider only the control flow of the process without regarding instructions. Context-sensitive and content-related analyses include those instructions. Goal-oriented analyses require additional information from the developer in which the developer describes the goals of the process.

Traditionally, process analyses focus on structural process properties, e.g., soundness [29]. The soundness property guarantees the absence of deadlocks in non-deterministic processes. We have developed a new approach to detect such deadlocks in conjunction with all the necessary information to repair them [15][16]. That approach has to be extended for the IR's additional language constructs. Although structural analyses consider only the control flow, they are suitable pre-processors for advanced analyses as they reduce the solution and failure space, c.f. SESE decomposition [32]. One has to show that it is possible to find further structural information, e.g., nodes with possible race conditions.

Since structural analyses can result in false-positive and false-negative analysis results [33], the consideration of data and instructions is essential to seriously support a process developer. However, less attention has been paid to process data and instructions in the literature. Sidorova [33] and our previous work [27][31] describe ways to include data in semantic analyses. Approaches of compiler theory can improve context-sensitive and content-related analyses by deriving predicate-logic expressions, by using path-sensitive data-flow analyses [34], by using instruction ordering techniques [35], or by using demand-driven approaches with backward traverses [36]. Especially, (C)SSA form is predestined for state space techniques since each variable is defined once and therefore the state space of a variable can be directly attached to it. All those approaches have to be reconsidered in the context of process analysis.

Goal-oriented analyses use approaches of AI planning. For this purpose, one has to define a precise *process domain* (properties which are in the focus of the process) that is used by the developer to describe the changes in this domain and the goal of the process. Then, the reachability of the goal can be verified. Furthermore, analyses can make suggestions to complete a process with respect to its goals. Our major focus lies on the adaption of AI planning techniques for the context of processes.

4) Error Handling: The error handler can improve the IR process by error correction and restructuring. The results of the semantic analyses are visualized and explained to the process's developer, and the error handler derives proposals for correction which then can be applied. To this end, the process has to be decomposed in such a way that failures can

be corrected locally without side effects.

The application of methods for automatic correction is not in the main line of research in BPM, resulting in only a handful of related approaches [37][38][39]. Most approaches consider preconditions of tasks within the process and how they can lead to deadlocks. Automatic correction then means to introduce weak conditions to avoid such deadlocks. Other methods derive a more general model of the process and afterwards construct a new process representing that model.

We follow another approach, in which the desired behavior of the process can never be completly derived from the process since the intention of the process is only in the mind of the developer. In this case, error handling has to interact with the developer to identify the best fitting solution. For this, approaches of AI planning are considered, which use basic rules to generate good solution proposals.

5) Coder and Annotator: If the process is correct, such that all verifiable properties hold, the coder and annotator enrich the IR with the results of the semantic analyses and afterwards possibly encode it into a mobile format. Both, the annotation of business processes as well as a mobile format, have not been in the focus of research. We want to consider approaches of compiler theory in which the annotation of programs (e.g., Java bytecode) or mobile formats are well understood. Our mobile format *SafeTSA*, for example, provides approaches to efficiently transform programs with a tree structure (AST) and (C)SSA form [40]. As mentioned before, each process can be represented by its PST and therefore it is possible to generate a mobile format, similar to SafeTSA, for business processes. In summary, we have to generate a SafeTSA conform mobile format for business processes to encourage their exchange.

B. Business Process Repository

After the process becomes executable, has been verified and possibly encoded into a mobile format, it can be stored within a business process repository. Thereby, the repository should provide features to find fitting process interaction partners by the use of its (semantic) annotations. For this purpose, promising proposals for methods [41][42] exist which can be extended and applied to the development of our system.

C. Consumer Side

As most steps of the consumer side are similar to those of the compiler, we want to only briefly discuss the engine in the following. On the consumer side, the business process is taken from the repository and is transformed back into the common IR (*decoder*). During that transformation, the process has to be verified once again (*verificator*) with the help of the annotated information, to guarantee a flawless transfer. As the results of the semantic analyses on the producer side have been annotated to the IR, that can be done fast.

Subsequently, an *interpreter* starts executing the process, for which Petri net-based or straight-forward (and almost sequential) approaches have been described in practice. However, we prefer to use a virtual machine as process engine since this approach is sufficient for Java. We imagine a main control unit loading the process and monitoring its execution. Furthermore, it starts a subprocess for each new control flow (e.g., after parallel branches) such that each control flow is handled by a seperate control unit with its own local memory, arithmetic logic unit, and input/ouput unit. The resulting architecture provides full parallelism and is able to execute sub processes on hetergenous subsystems (e.g., in a network). It can request services and other tools to support (user) tasks. The main control unit performs *dynamic semantic analyses* during the execution of a process to find runtime errors as early as possible. Since those analyses have full runtime information, indications of failure situations can be detected *before* they occur. A user has then the possibility to correct those failures with the help of a *runtime error handler*. Both, the dynamic semantics analyses and the runtime error handling mechanism, are based on the process's annotations, the analyser, and the error handling techniques of the compiler with the advantage of having full information about actual variable assignments.

IV. Conclusion and Future Work

In this paper, we have motivated the construction of a compiler, a common IR, a virtual machine, and detailed failure analyses for business process management. We have proposed a system which allows for the compilation, storing, and execution of processes based upon its own IR. That system uses state-of-the-art approaches and ideas from business process management, compiler construction, and artificial intelligence. There already exist approaches to realize such a system.

For the future, we recommend to develop and evaluate a compiler-based development and runtime environment for business processes without the consideration of data. Those environments should then be extended for processes with data. With this in mind, there are four main aspects being sequential considered: (1) an intermediate representation based on extended workflow graphs with regard to a virtual machine and its execution semantics, (2) process properties with static and dynamic analyses for their verification, (3) an error visualization, handling, and correction, and (4) process annotations for an efficient information transfer. The major goal is to show that such a business process management system is possible, applicable, and efficient. We are sure that such a system is the future of process development and will support processes for small and medium-sized enterprises and for the development of SOA applications.

References

[1] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, "Business process management: A survey," in Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings, ser. Lecture Notes in Computer Science, W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, Eds., vol. 2678. Springer, 2003, pp. 1–12.

[2] J. Koehler, T. Gschwind, J. M. Küster, H. Völzer, and O. Zimmermann, "Towards a compiler for business-it systems - A vision statement complemented with a research agenda," in Software Engineering Techniques - Third IFIP TC 2 Central and East European Conference, CEE-SET 2008, Brno, Czech Republic, October 13-15, 2008, Revised Selected Papers, ser. Lecture Notes in Computer Science, Z. Huzar, R. Kocí, B. Meyer, B. Walter, and J. Zendulka, Eds., vol. 4980. Springer, 2008, pp. 1–19.

[3] OMG, "Business Process Model and Notation 2.0," formal/2011-01-03, 2011, last access: February 18, 2015.

[4] A. Scheer, "Architecture of integrated information systems (ARIS)," in Information Infrastructure Systems for Manufacturing, Proceedings of the JSPE/IFIP TC5/WG5.3 Workshop on the Design of Information Infrastructure Systems for Manufacturing, DIISM '93, Tokyo, Japan, 8-10 November, 1993, ser. IFIP Transactions, H. Yoshikawa and J. Goossenaerts, Eds., vol. B-14. North-Holland, 1993, pp. 85–99.

 ISBN: 978-1-61208-387-2

[5] Alfresco, "Activiti BPM Platform," http://activiti.org/, last access: February 18, 2015.

[6] redhat, "jBPM - Open Source Business Process Management - Process engine," http://www.jbpm.org/, last access: February 18, 2015.

[7] IBM, "IBM WebSphere software - United States," http://www.ibm.com/software/websphere/, last access: February 18, 2015.

[8] AristaFlow, "AristaFlow - Aristaflow BPM Suite fr den BPM-Roundtrip in einem einzigen Werkzeug," http://www.aristaflow.com/bpmsuite.html, last access: February 18, 2015.

[9] Bonitasoft, "Bonitasoft - Open Source Workflow & BPM software," http://www.bonitasoft.com/, last access: February 18, 2015.

[10] M. L. Rosa, H. A. Reijers, W. M. P. van der Aalst, R. M. Dijkman, J. Mendling, M. Dumas, and L. García-Bañuelos, "APROMORE: an advanced process model repository," Expert Syst. Appl., vol. 38, no. 6, 2011, pp. 7029–7040.

[11] OASIS, Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, OASIS Std. 2, Rev. 0, apr 2007, last access: February 18, 2015.

[12] W. M. P. van der Aalst and A. H. M. ter Hofstede, "Yawl: yet another workflow language," Inf. Syst., vol. 30, no. 4, 2005, pp. 245–275.

[13] M. Weidlich, G. Decker, A. Großkopf, and M. Weske, "BPEL to BPMN: the myth of a straight-forward mapping," in On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 5331. Springer, 2008, pp. 265–282.

[14] W. Zhao, R. Hauser, K. Bhattacharya, B. R. Bryant, and F. Cao, "Compiling business processes: untangling unstructured loops in irreducible flow graphs," IJWGS, vol. 2, no. 1, 2006, pp. 68–91.

[15] T. M. Prinz and W. Amme, "Practical compiler-based user support during the development of business processes," in Service-Oriented Computing - ICSOC 2013 Workshops - CCSA, CSB, PASCEB, SWESE, WESOA, and PhD Symposium, Berlin, Germany, December 2-5, 2013. Revised Selected Papers, ser. Lecture Notes in Computer Science, A. Lomuscio, S. Nepal, F. Patrizi, B. Benatallah, and I. Brandic, Eds., vol. 8377. Springer, 2013, pp. 40–53.

[16] T. M. Prinz, N. Spieß, and W. Amme, "A first step towards a compiler for business processes," in Compiler Construction - 23rd International Conference, CC 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings, ser. Lecture Notes in Computer Science, A. Cohen, Ed., vol. 8409. Springer, 2014, pp. 238–243.

[17] J. de Bruijn, H. Lausen, A. Polleres, and D. Fensel, "The web service modeling language WSML: an overview," in The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings, ser. Lecture Notes in Computer Science, Y. Sure and J. Domingue, Eds., vol. 4011. Springer, 2006, pp. 590–604.

[18] W3C, OWL Web Ontology Language for Services, http://www.w3.org/Submission/2004/07/, World Wide Web Consortium W3C Std. 1, Rev. 0, nov 2004, last access: February 18, 2015.

[19] F. Henni and B. Atmani, "Dynamic web service composition. use of case based reasoning and AI planning," in Proceedings of the 4th International conference on Web and Information Technologies, ICWIT 2012, Sidi Bel Abbes, Algeria, April 29-30, 2012, ser. CEUR Workshop Proceedings, M. Malki, S. Benbernou, S. M. Benslimane, and A. Lehireche, Eds., vol. 867. CEUR-WS.org, 2012, pp. 22–29.

[20] H. Nacer and D. Aïssani, "Semantic web services: Standards, applications, challenges and solutions," J. Network and Computer Applications, vol. 44, 2014, pp. 134–151.

[21] M. Ghallab, D. S. Nau, and P. Traverso, Automated planning - theory and practice. Elsevier, 2004.

[22] W. Amme, T. S. Heinze, and J. von Ronne, "Intermediate representations of mobile code," Informatica (Slovenia), vol. 32, no. 1, 2008, pp. 1–25.

[23] W. M. P. van der Aalst, "The application of petri nets to workflow management," Journal of Circuits, Systems, and Computers, vol. 8, no. 1, 1998, pp. 21–66.

[24] S. Hinz, K. Schmidt, and C. Stahl, "Transforming BPEL to petri nets," in Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings, W. M. P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, Eds., vol. 3649, 2005, pp. 220–235.

[25] N. Lohmann, "A feature-complete petri net semantics for WS-BPEL 2.0," in Web Services and Formal Methods, 4th International Workshop, WS-FM 2007, Brisbane, Australia, September 28-29, 2007. Proceedings, ser. Lecture Notes in Computer Science, M. Dumas and R. Heckel, Eds., vol. 4937. Springer, 2007, pp. 77–91.

[26] J. Vanhatalo, H. Völzer, and J. Koehler, "The refined process structure tree," Data Knowl. Eng., vol. 68, no. 9, 2009, pp. 793–818.

[27] W. Amme, A. Martens, and S. Moser, "Advanced verification of distributed ws-bpel business processes incorporating cssa-based data flow analysis," International Journal of Business Process Integration and Management, vol. 4, no. 1, 2009, pp. 47–59.

[28] J. Lee, S. P. Midkiff, and D. A. Padua, "Concurrent static single assignment form and constant propagation for explicitly parallel programs," in Languages and Compilers for Parallel Computing, 10th International Workshop, LCPC'97, Minneapolis, Minnesota, USA, August 7-9, 1997, Proceedings, ser. Lecture Notes in Computer Science, Z. Li, P. Yew, S. Chatterjee, C. Huang, P. Sadayappan, and D. C. Sehr, Eds., vol. 1366. Springer, 1997, pp. 114–130.

[29] W. M. P. van der Aalst, A. Hirnschall, and H. M. W. E. Verbeek, "An alternative way to analyze workflow graphs," in Advanced Information Systems Engineering, 14th International Conference, CAiSE 2002, Toronto, Canada, May 27-31, 2002, Proceedings, ser. Lecture Notes in Computer Science, A. B. Pidduck, J. Mylopoulos, C. C. Woo, and M. T. Özsu, Eds., vol. 2348. Springer, 2002, pp. 535–552.

[30] W. Sadiq and M. E. Orlowska, "Analyzing process models using graph reduction techniques," Inf. Syst., vol. 25, no. 2, 2000, pp. 117–134.

[31] T. S. Heinze, W. Amme, and S. Moser, "A restructuring method for WS-BPEL business processes based on extended workflow graphs," in Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings, ser. Lecture Notes in Computer Science, U. Dayal, J. Eder, J. Koehler, and H. A. Reijers, Eds., vol. 5701. Springer, 2009, pp. 211–228.

[32] J. Vanhatalo, H. Völzer, and F. Leymann, "Faster and more focused control-flow analysis for business process models through SESE decomposition," in Service-Oriented Computing - ICSOC 2007, Fifth International Conference, Vienna, Austria, September 17-20, 2007, Proceedings, ser. Lecture Notes in Computer Science, B. J. Krämer, K. Lin, and P. Narasimhan, Eds., vol. 4749. Springer, 2007, pp. 43–55.

[33] N. Sidorova, C. Stahl, and N. Trcka, "Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible," Inf. Syst., vol. 36, no. 7, 2011, pp. 1026–1043.

[34] J. Fischer, R. Jhala, and R. Majumdar, "Joining dataflow with predicates," in Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2005, Lisbon, Portugal, September 5-9, 2005, M. Wermelinger and H. Gall, Eds. ACM, 2005, pp. 227–236.

[35] E. Duesterwald and M. L. Soffa, "Concurrency analysis in the presence of procedures using a data-flow framework," in Symposium on Testing, Analysis, and Verification, 1991, pp. 36–48.

[36] K. Winter, C. Zhang, I. J. Hayes, N. Keynes, C. Cifuentes, and L. Li, "Path-sensitive data flow analysis simplified," in Formal Methods and Software Engineering - 15th International Conference on Formal Engineering Methods, ICFEM 2013, Queenstown, New Zealand, October 29 - November 1, 2013, Proceedings, ser. Lecture Notes in Computer Science, L. Groves and J. Sun, Eds., vol. 8144. Springer, 2013, pp. 415–430.

[37] M. Gambini, M. L. Rosa, S. Migliorini, and A. H. M. ter Hofstede, "Automated error correction of business process models," in Business Process Management - 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings, ser. Lecture Notes in Computer Science, S. Rinderle-Ma, F. Toumani, and K. Wolf, Eds., vol. 6896. Springer, 2011, pp. 148–165.

 ISBN: 978-1-61208-387-2

[38] A. Awad, G. Decker, and N. Lohmann, “Diagnosing and repairing data anomalies in process models,” in Business Process Management Workshops, BPM 2009 International Workshops, Ulm, Germany, September 7, 2009. Revised Papers, ser. Lecture Notes in Business Information Processing, S. Rinderle-Ma, S. W. Sadiq, and F. Leymann, Eds., vol. 43. Springer, 2009, pp. 5–16.

[39] C. Wagner, “A data-centric approach to deadlock elimination in business processes,” in 3rd Central-European Workshop on Services and their Composition, Services und ihre Komposition, ZEUS 2011, Karlsruhe, Germany, February 21-22, 2011. Proceedings, ser. CEUR Workshop Proceedings, D. Eichhorn, A. Koschmider, and H. Zhang, Eds., vol. 705. CEUR-WS.org, 2011, pp. 104–111.

[40] W. Amme, N. Dalton, M. Franz, and J. von Ronne, “Safetsa: A type safe and referentially secure mobile-code representation based on static single assignment form,” in Proceedings of the 2001 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Snowbird, Utah, USA, June 20-22, 2001, M. Burke and M. L. Soffa, Eds. ACM, 2001, pp. 137–147.

[41] F. Klan and B. König-Ries, “A user-centered methodology for the evaluation of (semantic) web service discovery and selection,” in 4th International Conference on Web Intelligence, Mining and Semantics (WIMS 14), WIMS ’14, Thessaloniki, Greece, June 2-4, 2014, R. Akerkar, N. Bassiliades, J. Davies, and V. Ermolayev, Eds. ACM, 2014, p. 18.

[42] H. Si and Y. Zhao, “A structured p2p-based approach to semantic web services publication and discovery,” JSW, vol. 9, no. 7, 2014, pp. 1930–1940.

 ISBN: 978-1-61208-387-2