

Queensland University of Technology Brisbane Australia

This is the author's version of a work that was submitted/accepted for publication in the following source:

Van der Aalst, Wil, Dreiling, Alexander, Gottschalk, Florian, Rosemann, Michael, & Jansen-Vullers, Monique (2006) Configurable Process Models as a Basis for Reference Modeling. *Business Process Management Workshops (LNCS 3812)*, pp. 512-518.

This file was downloaded from: http://eprints.qut.edu.au/22487/

© Springer-Verlag

Notice: Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source:

http://dx.doi.org/10.1007/11678564_47

Configurable Process Models as a Basis for Reference Modeling – position paper –

W.M.P. van der Aalst^{1,3}, A. Dreiling^{2,3}, F. Gottschalk¹, M. Rosemann³, and M.H. Jansen-Vullers¹

¹ Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.

w.m.p.v.d.aalst@tm.tue.nl

² European Research Center for Information Systems, University of Münster Leonardo-Campus 3, 48149 Münster, Germany.

³ Queensland University of Technology, 126 Margaret St, Brisbane, QLD 4000, Australia.

Abstract. Off-the-shelf packages such as SAP need to be configured to suit the requirements of an organization. Reference models support the configuration of these systems. Existing reference models use rather traditional languages. For example, the SAP reference model uses Eventdriven Process Chains (EPCs). Unfortunately, traditional languages like EPCs do not capture the configuration-aspects well. Consider for example the concept of "choice" in the control-flow perspective. Although any process modeling language, including EPCs, offers a choice construct (e.g., the XOR connector in EPCs), a single construct will not be able to capture the time dimension, scope, and impact of a decision. Some decisions are taken at run-time for a single case while other decisions are taken at build-time impacting a whole organization and all current and future cases. This position paper discusses the need for *configurable process models* as a basic building block for reference modeling. The focus is on the control-flow perspective.

1 Introduction

The main objective of reference models is to streamline the design of particular models by providing a generic solution [19]. The application of reference models is motivated by the "Design by Reuse" paradigm. Reference models accelerate the modeling and configuration process by providing a repository of potentially relevant models. These models are ideally "plug and play" but often require some customization/configuration to be adjusted to individual requirements [7]. A configurable process model provides rules defining how a reference model can be adapted. Such a generating adaptation must be distinguished from non-generating adaptations as, e.g., aggregation, specialization or instantiation [5]. Unfortunately, the languages used for reference modeling [4, 8, 18] provide little or no support for configuration. The goal of this position paper is to discuss the need for *configurable process models*.

One of the most comprehensive models is the SAP reference model [8]. Its data model includes more than 4000 entity types and the reference process models cover more than 1000 business processes and inter-organizational business scenarios [19]. Most of the other dominant ERP vendors have similar or alternative approaches towards reference models. Foundational conceptual work for the SAP reference model has been conducted by SAP AG and the IDS Scheer AG in a collaborative research project in the years 1990-1992 [13]. The outcome of this project was the process modeling language Event-Driven Process Chains (EPCs) [13, 14], which has been used for the design of the reference process models in SAP. EPCs also became the core modeling language in the Architecture of Integrated Information Systems (ARIS) [21, 22]. It is now one of the most popular reference modeling languages and has also been used for the design of many SAP-independent reference models (e.g., the ARIS-based reference model for Siebel CRM or industry models for banking, retail, insurance, telecommunication, etc.). Despite its success, the basic EPC model offers little support for process configuration. It contains (X)OR connectors but it is unclear whether the corresponding decisions need to be taken at run-time (e.g., based on the stocklevel), at build-time (e.g., based on the size of the organization using SAP), or somewhere in-between (e.g., based on the period of the year or resource availability). Therefore, we developed the so-called *Configurable EPCs* (C-EPCs) [19, 9], a generic-monolithic approach for constructing re-usable models [10]. Indeed C-EPCs are extending the configuration opportunities of build-time operators [23, 20, 17]. However, they only provide a partial solution as they are only a representation variation, based on a specific language (EPCs), allowing the user to select or hide elements [5,6]. In this position paper we would like to trigger a discussion on requirements for configurable process models in a broader perspective.

The remainder of the paper is organized as follows. First, we elaborate on the concept of "choice" which is essential for configurable process models. Second, we approach the problem from a more theoretical viewpoint, i.e., we depict what the essence of configuration is. Finally, we briefly discuss Configurable EPCs as a first step towards such configurable models.

2 Configuration: It is all about making choices

This paper focuses on configurable process models, i.e., we restrict ourselves to the control-flow perspective [12]. There are many languages to model processes ranging from formal (e.g., Petri nets and process algebras such as Pi calculus) to informal (flow charts, activity diagrams, EPCs, etc.). Each of these languages provides some *notion of choice* (e.g., two transitions sharing a single input place in a Petri net or an (X)OR-split connector in an EPC). Typically, it is not possible to describe the nature of such a choice. At best one can either specify a Boolean condition based on some data element (data-based decision) or one can specify events that have to occur for triggering paths (event-based decision) [16]. The usual interpretation is that a choice is made at run-time, based on such a Boolean condition or based on occurring events. In the context of reference models, this interpretation is too narrow.

The *scope* of a decision can vary. For example, if a hospital uses a rule like "If a patient has high blood pressure a day before the planned operation, the operation will be canceled", then the scope of each choice (operate or not) is limited to a single patient. There may also be choices which affect more cases, e.g., consider the rule "If there is a major disaster in the region, all planned operations will be canceled." or also an entire process, e.g., "The admittance process requires patients to pre-register.". There may even be choices that affect all processes in some organizations. The classical process modeling languages, e.g., the languages used in workflow management systems [2, 12], allow only for one level of choices. Reference models have to allow for a broader spectrum of choices. Such choices are called configuration choices and are made at build-time. Configuration choices also affect choices at run-time. For example, at build-time one can choose not to use specific functionality offered by the system. Then no choice needs to be made at run-time anymore. But it may also be possible to use the functionality conditionally (e.g., depending on the workload). In this case the choice must be made at run-time. One can view configuration as *limiting* choices by making choices. Seen from this viewpoint, process modeling languages need to distinguish between run-time choices and configuration choices (i.e., at build-time). Note that the borderline between run-time choices and configuration choices may be a bit fuzzy as the following examples show.

- Based on the volume of the order, the goods are shipped by truck or mail.
- On Saturday, goods are shipped by truck.
- If stock is below 100 items, only preferred customers are serviced.
- The Dutch branches require a deposit, while this is not needed for branches in other countries.
- The organization chooses not to allow for pre-shipments.

Each of these choices is at another level. However, the processes in e.g. the SAP reference model show only one type of choice: the (X)OR-split connector. This triggered us to develop the so-called C-EPCs.

3 Configuration: A theoretical perspective

As described above a reference model provides a generic solution that needs to be configured for a specific situation. A generic-monolithic approach for model reuse should guide the user to a solution fitting to the individual requirements [10]. Therefore the reference model must be able to provide a complete, integrated set of all possible process configurations. This means the reference model is the least common multiple of all process variations, which leads to *inheritance* of dynamic behavior [1,3]. A reference model can be seen as a subclass of all concrete models. A concrete model itself is a superclass of the reference model. This may create confusion as the term "super" is intuitively connected to the bigger and at first existing reference model (e.g., in [24] traditional inheritance was altered to depict the reference model as superclass). However, it corresponds to the traditional notion of inheritance in which the subclass adds things to the superclass (e.g., additional methods or attributes). So configuration can be described as the reverse of inheritance. This allows us to use some of the ideas described in [1,3], in particular we use the idea of *hiding* and *blocking*.



Fig. 1. Three labeled transition systems: (a) the initial model (e.g., the reference model), (b) a particular configuration hiding and blocking specific edges/labels, and (c) the resulting model.

Any process model having formal semantics can be mapped onto a labeled transition system. The nodes in a labeled transition system represent states, the directed edges represent transitions, and each transition has a label denoting some event, action or activity. Traditional choices in the process model, correspond to nodes in the labeled transition system with multiple output arcs. Consider Figure 1(a) showing a labeled transition system. In the initial state (the top node, edges go from top to bottom) there is a choice between a and b. If a is selected, the next step is c and then there is a choice between d and e, etc. If we consider Figure 1(a) to be a reference model, a configuration of this model should select the desired parts. This can be done by blocking and hiding edges or labels. In Figure 1(b) one edge is blocked and three edges are hidden. Hiding and blocking should be interpreted as in [1,3], i.e., hiding corresponds to abstraction and blocking corresponds to encapsulation. If an edge is blocked, it cannot be taken anymore. By hiding an edge the path is still possible but the associated label is no longer relevant, i.e., it is renamed to a silent step τ . One can think of the latter as simply skipping the edge. Figure 1(c) shows the resulting model after blocking and hiding the edges indicated in Figure 1(b).

A configurable process model should allow for the specification of which edges/labels can be blocked and hidden/skipped. An interesting question is whether it should be possible to defer this decision to run-time. In the latter case, there would be two more options: *optional blocking* and *optional hiding* (to be decided at run-time).

4 Configuration: An example of a language

To conclude this position paper we introduce *Configurable EPCs* (C-EPCs) as an example for a configurable process modeling language. C-EPCs are an extension of the classical EPCs [13]. A classical EPC consists of functions (i.e., the activities), events and connectors. Functions follow events and events follow functions. Moreover, to model splits and joins in a process connectors may be used. There are three types of connectors: AND, OR and XOR. AND-splits and AND-joins may be used to model parallel routing. XOR-splits and XOR-joins may be used to model the selection of specific routes (e.g., an "if then else" construct). OR-splits and OR-joins may be used to model a mixture of conditional and parallel routing. (However, the semantics of the OR-join is still debated [14].)

In a C-EPC both functions and connectors may be configurable. Configurable functions may be included (ON), skipped (OFF) or conditionally skipped (OPT). Configurable connectors may be restricted at build-time, e.g., a configurable connector of type OR may be mapped onto an AND connector. Local configuration choices like skipping a function may be limited by configuration requirements. For example, if one configurable connector c of type OR is mapped onto an XOR connector, then another configurable function f needs to be included. This configuration requirement may be denoted by the logical expression; $c = OR \Rightarrow f = ON$. In addition to these requirements it is possible to add guidelines, supporting the configuration process.

Figure 2 shows a C-EPC describing an invoice verification process. The classical EPC is extended with configurable functions and connectors (indicated using thick lines). For example function *Invoicing Plan Settlement* is configurable, i.e., it may be included (ON), skipped (OFF) or conditionally skipped (OPT). The diagram shows also some configurable connectors. In this position paper we do not further elaborate on C-EPCs. For more information, we refer to [19,9]. The important thing to note is that it is possible to extend a language like EPCs with configurable elements. Moreover, there are two types of choices: (1) configuration choices made at build-time and (2) "normal" choices made at run-time.

C-EPCs can be seen as a rather naive, but very intuitive, configuration language that allows (optionally) blocking and hiding of edges/labels at build-time for specifying the configuration of the model. Using the theory developed in [1, 3] and basic notions such as simulation, bisimulation, and branching bisimulation [11, 15] on the one hand and practical experiences using C-EPCs on the other hand, we hope to develop more mature configuration languages.

The aim of this position paper is to trigger a discussion on configurable process models. To do this we argued that configuration is strongly related to



Fig. 2. A Configurable EPC.

the timing and scope of choices. We also showed an example of a language (C-EPCs). However, to allow for a more language-independent discussion we also tried to capture the essence of configuration in terms of (optional) hiding and blocking of edges or labels.

References

- W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125-203, 2002.
- W.M.P. van der Aalst and K.M. van Hee. Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA, 2002.
- T. Basten and W.M.P. van der Aalst. Inheritance of Behavior. Journal of Logic and Algebraic Programming, 47(2):47–145, 2001.
- 4. J. Becker, M. Kugeler, and M. Rosemann, editors. *Process Management: A Guide for the Design of Business Processes.* Springer-Verlag, Berlin, 2003.
- J. Becker, P. Delfmann, R. Knackstedt. Konstruktion von Referenzmodellierungssprachen: Ein Ordnungsrahmen zur Spezifikation von Adaptionsmechanismen für Informationsmodelle. In WIRTSCHAFTSINFORMATIK, 46(2004)4, pages 251–264.
- J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, D. Kuropka. Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In Proceedings of the 15th Information Resources Management Association International Conference. New Orleans, 2004.

- 7. P. Bernus. Generalised Enterprise Reference Architecture and Methodology, Version 1.6.3. IFIPIFAC Task Force on Architectures for Enterprise Integration, 1999.
- T. Curran and G. Keller. SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Upper Saddle River, 1997.
- A. Dreiling, M. Rosemann, W.M.P. van der Aalst, W. Sadiq, and S. Khan. Modeldriven process configuration of enterprise systems. In O.K. Ferstl, E.J. Sinz, S. Eckert, and T. Isselhorst, editors, Wirtschaftsinformatik 2005. eEconomy, eGovernment, eSociety, pages 687–706, Physica-Verlag, Heidelberg, 2005.
- P. Fettke and P. Loos. Methoden zur Wiederverwendung von Referenzmodellen

 Übersicht und Taxonomie. In J. Becker, R. Knackstedt, editors, Referenzmodellierung 2002: Methoden Modelle Erfahrungen, Arbeitsberichte des Instituts
 für Wirtschaftsinformatik Nr. 90 (in German), pages 9–33. University of Münster,
 Münster, 2002.
- R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. In Journal of the ACM, 43(3):555–600, 1996.
- S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, London, UK, 1996.
- G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.
- E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In J. Desel, B. Pernici, and M. Weske, editors, International Conference on Business Process Management (BPM 2004), volume 3080 of Lecture Notes in Computer Science, pages 82–97. Springer-Verlag, Berlin, 2004.
- R. Milner. A Calculus of Communicating Systems, volume 92 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1980.
- 16. M. Owen and J. Raj. BPMN and Business Process Management Introduction to the New Business Process Modeling Standard, Popkin Software, 2003.
- M. Rosemann. Komplexitätsmanagement in Prozessmodellen: methodenspezifische Gestaltungsempfehlungen f
 ür die Informationsmodellierung (in German). Gabler, Wiesbaden, 1996.
- M. Rosemann. Application Reference Models and Building Blocks for Management and Control (ERP Systems). In P. Bernus, L. Nemes, and G. Schmidt, editors, Handbook on Enterprise Architecture, pages 596–616. Springer-Verlag, Berlin, 2003.
- M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. In Information Systems (to appear, also available from BPMCenter.org), 2005.
- 20. M. Rosemann and R. Schütte. Grundsätze ordnungsmäßiger Referenzmodellierung. In J. Becker, M. Rosemann, R. Schütte, editors, Entwicklungsstand und Perspektiven der Referenzmodellierung, Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 52 (in German), pages 16–33. University of Münster, Münster, 1997.
- 21. A.W. Scheer. Business Process Engineering, Reference Models for Industrial Enterprises. Springer-Verlag, Berlin, 1994.
- 22. A.W. Scheer. ARIS: Business Process Modelling. Springer-Verlag, Berlin, 2000.
- R. Schütte. Grundsätze ordnungsmäßiger Referenzmodellierung Konstruktion konfigurations- und anpassungsorientierter Modelle (in German). Gabler, Wiesbaden, 1998.
- 24. A. Schwegmann. Objektorientierte Referenzmodellierung: theoretische Grundlagen und praktische Anwendung (in German). Gabler, Wiesbaden, 1999.