Realization of Business Processes

Andreas Speck, Andreas Rusnjak, Marcel Schulz

aspe@informatik.uni-kiel.de
aru@informatik.uni-kiel.de
marcel.schulz@intershop.de

Abstract: Business process models are the fundamental models of commercial systems. Therefore the business processes are issue of optimization as well as verification. When a business process has been identified as comparatively good (or optimal) it is of interest if the realization of the system still realizes this process. In this paper we present an approach how to verify such realizations. As application system we use the e-commerce system Intershop Enfinity which provides an executable process model: the *Pipeline Model*. These *Pipeline Models* have to be transferred to formal automata models which then may be checked. As tools to check we use model checkers.

1 Introduction

A main concern for the optimization of business processes is the realization of business processes. This means that it is desirable that a comparatively good (or optimal) model of a business process is also realized by the implementation of the system. The other way round it may also be of interest to know if an implementation does not meet the specified process model. In this case it may be analyzed if there is at all a chance to realize the optimized business process model.

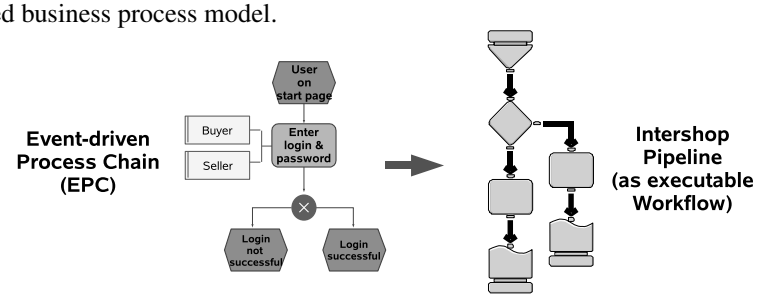

Figure 1: EPC Model and *Pipeline Model*

Due to the lack of standardized rules there are almost infinite possibilities for the realization of business processes. Therefore, we need to select a specific methodology or technology to implement business processes. In our case we focus on executable models, or to be more precise on executables workflow models. A well-known example for such an executable workflow model may BPEL4WS modeling the execution order of web services or the proprietary *Pipeline Model* of Intershop Enfinity [Bre02].

Similarely to BPEL4WS the *Pipeline Model* connects executable services (here named *Pipelets*). The *Pipelets* as nodes in the *Pipeline Model* may trigger the execution of comparatively large service (realized as EJBs). The differences between the *Pipeline Model* a BPEL4WS are: *Pipeline Models* do not model the interaction of distributed services like web services. Usually *Pipeline Models* focus on the interaction within one server. *Pipeline Models* are proprietary models limited to the e-commerce system Intershop Enfinity. The *Pipeline Models* are executed by a modified version of the Tomcat application server being part of the Intershop Enfinity system. The *Pipeline Models* were introduced before the web services and BPEL4WS were invented. *Pipeline Models* are part of the ARIS [Sch98] profile *ARIS for Enfinity* [Bre02].

2 EPC Model and Pipeline Model

The basic elements of an EPC model are shown in figure 1 (on the left): The control flow is symbolized by a sequence of events (magenta hexagons) and functions (green rectangles with rounded edges) which are connected by arrows representing the control flow. Branches in the control flow are defined by the Boolean logic operators: AND, OR and XOR. EPC models may be applied to express requirements of systems, specifically regarding the temporal behavior and they are also used in the design phase in order to meet the requirements. EPC models themselves may also be issue of verification [FSRK05].

The proprietary Intershop *Pipeline Models* (c.f. figure 1 on the right) describe actions of an e-commerce system and the paths linking these actions. The actions are represented by *Pipelets* which are the rounded rectangles. *Pipelets* consist of a declaration (in XML) and a definition in Java code which is then executed at run-time. This makes the model executable. The links between the *Pipelets* are the control flow represented by arrows and branching elements (as depicted in figure 1) or loop elements. The *Pipelines* are specifically used for web-based systems which mean that the *Pipelines* describe the activities between web pages presented to the clients. The *Pipelines* model and realize the interaction between two web pages. Therefore, the *Pipeline Models* provide modeling elements (Start element at the top of the *Pipeline Model*) for an ingoing interaction initiated by a clients action on a web page (e.g. activating a link or initiating a login action) and the outgoing interaction (Template Interaction, the two model elements at the bottom) which the feed the web page presenting the *Pipelines* results (in case of the example in figure 1: successful or unsuccessful login).

3 Formalization and Checking of Pipelines

In general the transformation of the Intershop *Pipeline Models* may be transformed straight forward into automata representation. Figure 2 shows these two formats. The example used is a price alert function. The function checks if a specific price goes beyond a defined threshold. If this condition is true (`PriceAlert = true` and additionally `PriceError = false`) then a mail is sent. This checking procedure is usually performed in a loop. In case the price doesn't change this is considered as `PriceError`

which means that the condition `PriceError = true` is set. This transformation into an automaton representation allows to use checking technology to verify automatically the model. The technology we have chosen is model checking [CGP01]. Model checking uses temporal logic in order to express the temporal sequence of states (the *Pipeline Model* elements are transformed to these states). Temporal logic provides besides the known Boolean logic operators temporal operators to express the temporal order. In our case we use the *Computational Tree Logic* (CTL) [BBF+01].

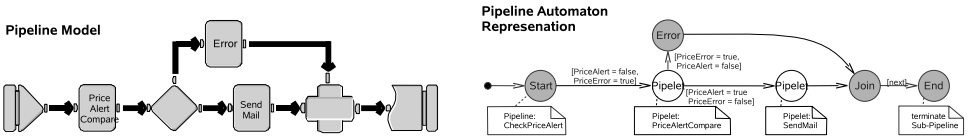

Figure 2: Example *Pipeline Price Alert* as *Pipeline Model* and Automaton Representation

The functionality of the *Pipeline Price Alert* from above may be expressed with CTL as follows: $A\,[(PriceAlert = \bot)\ W\ SendMail]$.

The formula expresses that always as long as the *Price Alert* is not true the *Pipelet SendMail* will not be sent.

4 Modeling and Checking of Real Systems

The examples up to now are quite small. Real e-commerce systems derived from the Intershop Enfinity base consist usually of several thousand *Pipelines* and *Sub-Pipelines* which are build of ten to hundred thousands of *Pipelets*.

Nevertheless these systems realize the functionality modeled in business processes. It is highly desirable to find an aid for (semi-) automated checking of these systems.

In order to illustrate the number of *Pipelines* and *Pipelets* we still concentrate of a small aspect of an e-commerce system: the presentation of products. This is one of the main functionality and usually most present to the customers using a web shop system. Usually this presentation functionality is always the way to present products the customers got arbitrarily, as opening presentation or as a result of concrete search.

Figure 3 shows the sample presentation in an ordinary web browser (Microsoft Internet Explorer in this case). Actually this is not a real web shop but a research system to examine the functionality.

The automaton in figure 3 consists of more than 50 nodes. Moreover, not all *Sub-Pipelines* are considered in this automaton. What is considered additionally is to the control flow of the web page template. This template is transformed by a specific processor to a simple HTML-page according to the number of the products to be displayed. Actually the automaton of figure 3 is only a sub-automaton of the large automaton representing the entire system. When we have a look at this sub-automaton it becomes clear that the checking of

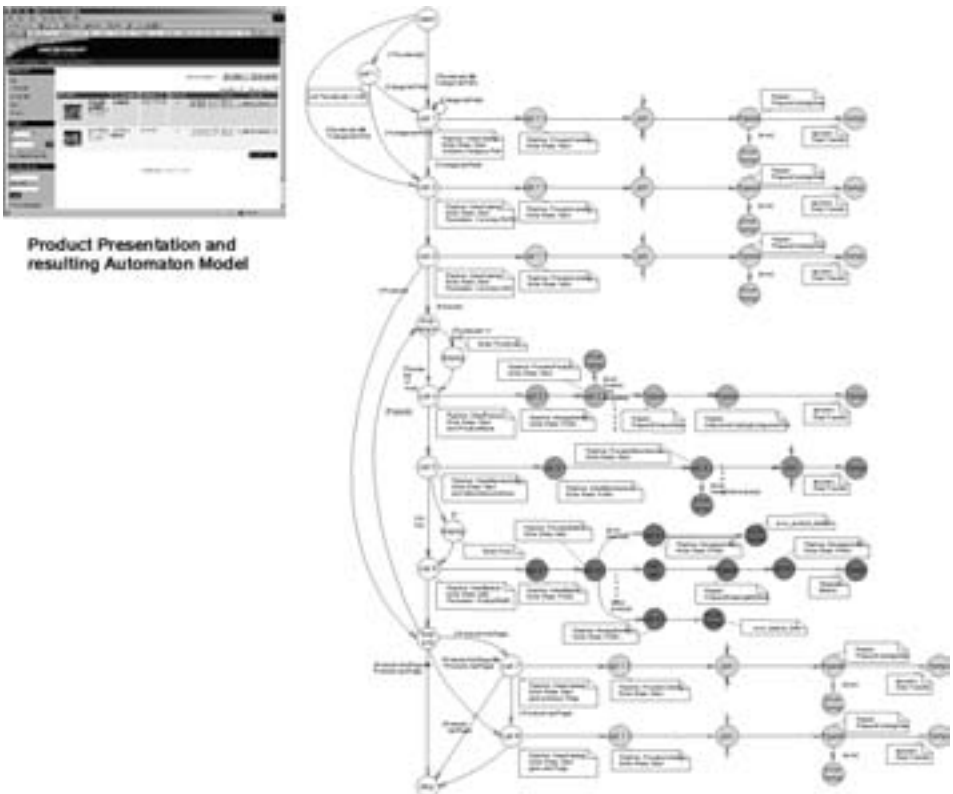

Figure 3: Product Presentation Web Page of Intershop Enfinity

such an automaton by hand is not that easy than the first examples. And if we take into account the number of several ten-thousand nodes of the complete automaton of a (still small) realization of a web shop system the need for automated support for the checking of such a system is obvious.

5 Related Work

The verification approach we use is based on a push-button formal technique to verify the fulfillment of automata-based specifications which is model checking [CGP01]. It uses (temporal) logics as specification language and checks these (temporal) requirements against a model. As opposed to other formal approaches (e.g. theorem provers) it is more restricted in what can be verified leading, on the other hand, to the advantage of a developer-friendly extensive automatization. The verification of software models has been a key issue in the model checking research for a long time (as in [EC80] or [McM93]).

In [FPR06] the verification of workflow-based systems is presented. This paper intro-

duces some of the basic principles we rely on in our paper. Further research on the verification of business process systems is the evaluation of different technologies in the area of model checking and other formal techniques in order to solve specific problems. Such problem-specific solutions have been elaborated. A further alternative approach is to apply constraint satisfaction presented in [Run09]. This paper concentrates on the configuration during design time. The rules for the checking and specifically the reuse of these rules are important issues. [Ger04] discusses the possibilities for reusing the specification rules.

An overview about the application of the proposed checking technology may be found at [FSRK05]. Here a first approach is presented. However, this approach focuses on the verification of the business process models (EPCs) itself.

Other approaches for the verification of business process systems are based on Petri nets (e.g. [DBS08] using BPMN). With Petri nets the business processes are mapped to the Petri net elements similar to our approach of direct mapping. In the Petri net based approaches the verification is often based on bi-simulation and algebraic solutions (e.g. [Mor08]).

6 Conclusion and Future Work

The presented approach allows verifying executable workflow models. As example we use the proprietary Intershop *Pipeline Models*. The rules applied are derived from best practices or business rules as they exist for almost any commercial system. Moreover, these rules for the executable model may be the optimized business processes themselves.

The paper demonstrates a straight-forward approach to transform executable workflow models (Intershop *Pipeline Models*) into automata which are the base of a formal verification. As verification technology we apply model checking. Our proposal allows checking comparatively large models. Comparatively means large from a human perspective. For the model checking tool the number of states is no real problem as we do not express critical items like numbers in the state space. The automated checking is provides the advantages of any automated process: It reduces the efforts for the human quality assurance teams while the number of errors is decreased.

The current state of the approach still uses the temporal logic CTL. Although we empirically learned that the application of this kind of logic seems to be very easy especially for quality assurance personal this will be issue of an improvement by presenting a more intuitive graphical notation. We may take the example of [FF08]. Here a graphical representation of CTL is introduced.

As we already stated the straight-forward transform results in simple but still models of a size processable by a model checker. A more elaborated concept is the differentiation of the element types as proposed in [Pul09] which may be adapted to our work.

This reduces the state space of the checking and allows to get a better feed-back when we analyze the checking results.

References

[BBF+01] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification – Model-Checking Techniques and Tools*. Springer, Berlin, Germany, 2001.

[Bre02] M. Breitling. Business Consulting, Service Packages & Benefits. Technical report, Intershop Customer Services, Jena, 2002.

[CGP01] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts; London, England, 3 edition, 2001.

[DBS08] M. De Backer and M. Snoeck. Business Process Verification: a Petri Net Approach. Technical report, Catholic University of Leuven, Belgium, 2008.

[EC80] E. A. Emerson and E. M. Clarke. Characterizing Correctness Properties of Parallel Programs Using Fixpoints. In *ICALP 1980, Automata, Languages and Programming, 7th Colloquium*, pages 169–181. Springer LNCS 85, 1980.

[FF08] S. Feja and D. Fötsch. Model Checking with Graphical Validation Rules. In *15th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS 2008), Belfast, NI, GB*, pages 117–125. IEEE Computer Society, April 2008.

[FPR06] D. Fötsch, E. Pulvermüller, and W. Rossak. Modeling and Verifying Workflow-based Regulations. In *Proceedings of the Workshop on Regulations Modelling and their Validation & Verification (REMO2V)*. Namur University Press, June 2006.

[FSRK05] D. Fötsch, A. Speck, W. Rossak, and J. Krumbiegel. A Concept of Modelling and Validation of Web based Presentation Templates. In *17. Internationale Tagung Wirtschaftsinformatik 2005 (WI2005)*, pages 391–406. Physika Verlag, February 2005.

[Ger04] R. Gerrits. Business rules, can they be re-used? *Business Rules Journal*, 5, 2004.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[Mor08] S. Morimoto. A Survey of Formal Verification for Business Process Modeling. In *ICCS 2008, 8th International Conference*, pages 514–522. Springer LNCS 5102, 2008.

[Pul09] E. Pulvermüller. Reducing the Gap between Verification Models and Software Development Models. In *The 8th International Conference on Software Methodologies, Tools and Techniques (SoMeT 2009)*, pages 297–313. IOS Press, 2009.

[Run09] W. Runte. Modelling and Solving Configuration Problems on Business Processes Using a Multi-Level Constraint Satisfaction Approach. In *The Young Researchers Workshop on Modeling and Management of Business Processes (YRW-MBP 2009)*, pages 237–238. GI LNI 147, 2009.

[Sch98] A.-W. Scheer. *ARIS - Modellierungsmethoden, Metamodelle, Awendungen.* Springer, Berlin, Germany, 1998.