Institute of Architecture of Application Systems

Business Process Change Management based on Process Model Synchronization of Multiple Abstraction Levels

Monika Weidmann[1], Modood Alvi[1], Falko Koetter[1], Frank Leymann[2], Thomas Renner[1], David Schumm[2]

[1] Fraunhofer Institute for Industrial Engineering IAO
Stuttgart, Germany

[2] Institute of Architecture of Application Systems,
University of Stuttgart, Germany

BibTeX

```
@inproceedings{WeidmannAK11,
  author    = {Monika Weidmann and Modood Alvi and Falko Koetter
               and Frank Leymann and Thomas Renner and David Schumm},
  title     = {Business Process Change Management based on Process Model
               Synchronization of Multiple Abstraction Levels},
  booktitle = {Proceedings of SOCA 2011},
  year      = {2011},
  publisher = {IEEE Computer Society}
}
```


Universität Stuttgart
Germany

Business Process Change Management based on Process Model Synchronization of Multiple Abstraction Levels

Monika Weidmann[1], Modood Alvi[2], Falko Koetter[1], Frank Leymann[2], Thomas Renner[1], David Schumm[2]
[1]Fraunhofer Institute for Industrial Engineering (IAO) Stuttgart, Germany firstname.lastname@iao.fraunhofer.de
[2]Institute of Architecture of Application Systems (IAAS) University of Stuttgart, Germany
firstname.lastname@iaas.uni-stuttgart.de

***Abstract*—Management of business processes is typically performed on multiple levels, each with different granularity, language constructs, and abstraction. Starting from an initial sketch of the activities to be performed, several refinements are made to entirely specify the business process, its artifacts, and participants. Then, information relevant for process execution can be added to enable efficient automation in the context of a service-oriented architecture (SOA). However, dealing with changes initiated by business or technology is a key difficulty in this approach. If change management is not performed properly then process models become out of sync which results in losing the alignment of business and IT. To address this challenge, we propose a synchronization method based on model element correspondence that considers change management between process models on different abstraction levels. We show how synchronization can be established and changes are propagated using a change queue for synchronization continuity. Finally we present a prototypical implementation of the key concepts.**

Keywords-process BPM; model synchronization; abstraction levels; change propagation; business process modelling;

I. INTRODUCTION

Nowadays, many companies make significant investments in business process improvement [1]. Business processes are available in companies in different levels of detail and on different levels of abstraction, as this is for example required for maturity models like CMMI [1]. The introduced business process models are used for various scenarios, starting from the pure documentation of business processes to automated execution of business processes based on service orchestration through a workflow engine in the context of a service-oriented architecture (SOA) [2].

Due to fast changing market conditions business processes also need to be quickly and flexibly adaptable [3]. In practice, these requirements arise and the corresponding changes are made, but often (or typically) they are not propagated to the process models on other levels of abstraction. Other reasons for unsynchronized process models are on the one hand a different project-driven creation time of partly overlapping process models and on the other hand different creation purposes, like showing one individual aspect of a business process, not cross-checking with other existing process models. Typically there is no alignment phase between the existing higher and lower level models. This problem is known today and as well in the past as the *alignment problem* [4][5].

As a consequence, there is a strong need for a synchronization mechanism and change management functionality for process models on different levels of abstraction. We introduce model correspondence between process models on different levels of abstraction, which allows propagating changes top-down and bottom-up. Further, we contribute the concept of change queues which are used to guide the user through the synchronization procedure that is based on change operations.

This paper is structured as follows. In Section II we discuss state of the art research on which we build to provide the fundamentals for our approach. Section III introduces a running example. In Section IV, we describe the synchronization concept. In Section V we describe the prototype we implemented to show-case the major concepts of the approach. Finally, in Section VI we give a conclusion and provide an outlook on future work.

II. RELATED WORK

In this section, we discuss work related to this topic. We consider mainly three branches of research: (1) process model alignment, (2) similarity measures and difference identification as well as model-driven development in/for process models, and (3) change patterns in processes.

Given a pair of business process models, the approach described in [6] compares graph-based and lexical alignment techniques. In [7] a business-IT-mapping model is introduced, which defines detailed mapping types to map business process activities to technical process activities. [2] introduces realization types in order to define the relations between business process steps and technical process steps based on services. A similar approach is undertaken in [8] by introducing a “process support layer” to bridge the gap between process models and enacted models. Admitting the ambiguity of automatic lexical alignment [9] enriches business process models with ontology and matches activities. All these approaches have in common that the mapping of elements of different models is either automatically or semi-automatically derived, though using different techniques. We suppose that the models on different abstraction levels differ enough that a fully-automatic mapping is not applicable. However, the algorithms can be used to propose a mapping in order to

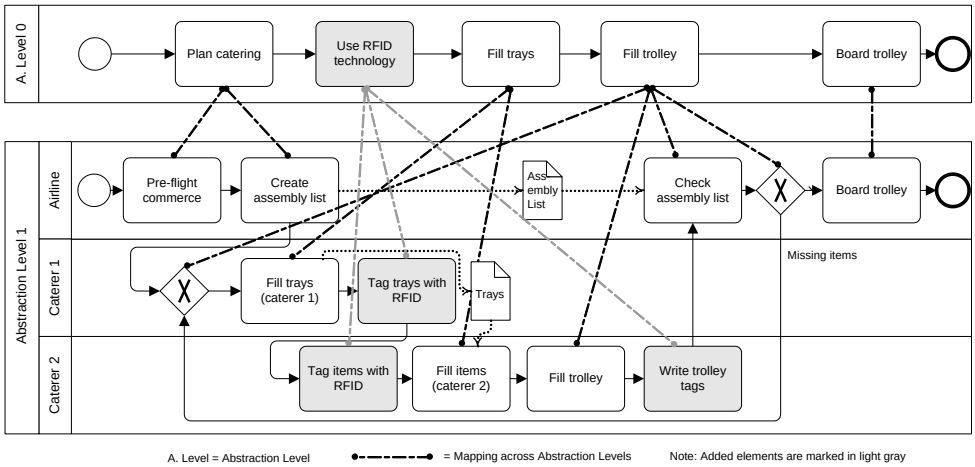

Figure 1. Running example – An airline catering process which is changed in order to make use of RFID technology

guide the user when initializing the mapping, given existing process models on different levels of abstraction.

Regarding similarity measures, the work of [10] derives a process structure tree [11] in order to find and resolve differences in process models. One work introduces a normal form for detecting semantically similar process fragments [12]. Another work which is in search of differences in process models is [13], which uses formal semantics to distinguish eight types of possible differences. Other approaches define measures on process similarity based on abstracted behavior [14] or semantic similarity [15]. In [16] business process models are abstracted into behavioral profiles in order to identify change regions and propagate changes. In the business IT alignment scenario, process models may indeed differ from one abstraction level to another and is not automatically computable. Thus, the importance of model similarity for our work lies in identifying the differences of one and the same process model resulting from change.

Process model change patterns are frequently applied forms of model changes such as adding or moving an activity in a process. In this field of research on process models [17] gives a well-structured overview of change operations in process models. A formal basis has been introduced in [18]. We rely on the change operations as fundamental basis for defining change propagation mechanisms on corresponding process elements.

III. RUNNING EXAMPLE

In this section, we introduce an abbreviated and strongly simplified version of an airline catering process developed in another research project [19] in BPMN 2.0 [20]. On the highest abstraction level, Level 0, the rough version of the process is modeled conceptually (see top part of Figure 1). On the lower abstraction level 1 (see lower part of Figure 1) this process is modeled in more detail. We use the concept of pools in order to show which process belongs to which abstraction level. The thick dashed lines with connectors are not part of BPMN 2.0, but show how the corresponding process elements are mapped across the abstraction levels. The gray elements show the changes which will be applied in the running example.

IV. SYNCHRONIZATION CONCEPT

In this section, we introduce the concepts to support change management through process synchronization. An overview of the concept is shown in Figure 2. When a certain change is made in one abstraction level, this change operation is logged in the *change queue* of this level. This change is propagated to all models which are *synchronized* with the process on level 0, which means they are subscribed for change information. In the example in Figure 2, we consider top-down propagation only, though the approach also supports bottom-up and middle-out propagation. For synchronization, the user on level 1 needs to reflect the *changes* made in level 0 in the level 1 process model by adding three activities. The goals of the introduced concept are to (1) ensure *synchronization* of all changes, (2) support of various mapping types of business processes across different levels (1:1, 1:n), (3) support various mapping types across multiple modeling elements on different abstraction levels (1:1, 1:n, n:1), and (4) introduce a synchronization concept for *bottom-up* and *top-down* change propagation based on user decisions

An ***abstraction level*** is a well-established concept in business process management practice and research, usually with the goal to bridge the gap between business and IT [21][22]. In our approach, an abstraction level *L* is a set of

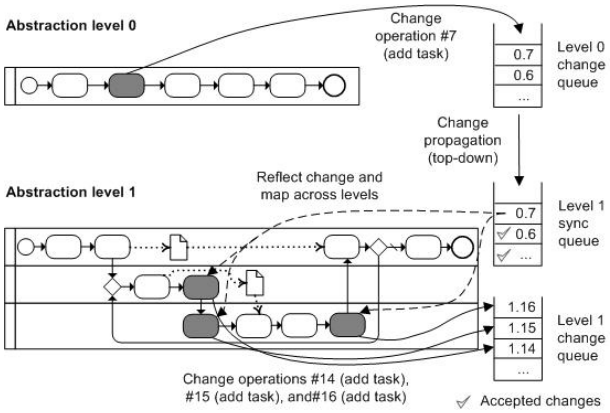

Figure 2. Overview of concept: Changed are logged in a change queue which is used to guide the user in reflecting these changes accordingly

process models, modeled with a level-specific subset of process modeling elements of the used process metamodel. Abstraction levels are numbered in order of detail with $k = 0$ as the highest-level process model.

We define a ***model correspondence*** mapping between two business process models on neighboring abstraction levels k and $k + 1$ as follows: Given two process graphs P_1 , P_2 where

(1) $P = (A, E, T_A, T_E, t_A, t_E)$, where
A is the set of activities,
$E \subseteq A \times A$ is the set of edges,
T_A is the subset of activity types available on the current level (activity types include gateways),
T_E is the subset of edge types available on the current level,
$t_A : A \rightarrow T_A$ maps each activity to an activity type,
$t_e : E \rightarrow T_E$ maps each edge to an edge type

we define the model correspondence mapping Z between P_1 and P_2 as

(2) $Z = (P_1, P_2, R, U)$, where
$R \subseteq (A_1 \times A_2) \cup (A_2 \times A_1)$ is the set of correspondences between levels
$U \subseteq (A_1 \cup A_2)$, $U = \{a \in (A_1 \cup A_2) \mid \forall x \in (A_1 \cup A_2): !\exists(a, x) \in R \wedge !\exists(x, a) \in R\}$ is the set of activities for which no correspondences have been defined.

In the running example in Figure 1 at the beginning ignoring the gray parts no activities without correspondence exist, U is empty, all activities are matched. The set of mappings R contains for example ("Plan catering", "Pre-flight commerce").

The definition above allows for one higher-level model to be associated to one or more lower-level models. It is common to split processes with increasing detail (see Figure 3).

We propose handling ***change management*** by the introduction of change queues and an algorithm supporting the propagation of changes (see sketch in Figure 2). Our approach builds on the 14 change operations introduced in [17]. Additionally, we consider two novel change operations *split* and *merge* of activities. We distinguish two operation types: *simple change operations* are addition or deletion of an activity or gateway or edge, whereas *complex change operations* like moving an activity use a combination of simple change operations. For a subset of exemplary complex change operations the operation and its effect on the model correspondence mapping have been defined in detail:

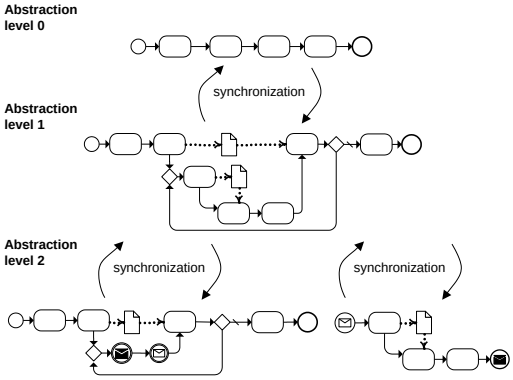

Figure 3. Mapping across abstraction levels and multiple lower-level processses

Addition of activity *x*
User adds activity x to process graph P_k
Precondition: $x \notin A_k$
Mapping change: $U := U \cup \{x\}$
Set of changed activities: $V = \{x\}$
Protocol: activity x has been added
Postcondition: $x \in A_k \wedge x \in U$ for all correspondence mappings of P_k

Swap activities x and y
User swaps two activities x and y in process graph Pk
Precondition: $x \in Ak \wedge y \in Ak$
Mapping change: none, but lower level activities might have to be swapped as well
Set of changed activities: V = {x, y}
Protocol: activity x and y have been swapped
Postcondition: $x \in Ak \wedge y \in Ak$

The three change operations deletion, move and replace have been defined in detailed, but omitted for reasons of space. All change operations performed on a level k have to be propagated to neighboring levels. To achieve this, we define a change queue CQ_k for each level k. Each change queue is an ordered list of change operations. Whenever a change operation is performed on a model P_k, it is appended to CQ_k. After changes are performed on one level, they have to be synchronized with the other levels. To achieve this, each change operation in the change queue has to be propagated to all neighboring levels. We introduce the concept of *sync queues* $SQ_{k,k+1}$ and $SQ_{k,k-1}$. A sync queue is the change queue propagated to the neighboring levels.

Additionally it is possible to enhance the change operations with "recommended corresponding operations". In any case, each element without correspondence (contained in U) has to be corresponded to an element on the current level, as we want the users to be notified of every change. For such cases, on the lower level one could define an activity and comment it with the measures taken. After correspondence is defined, a change is *accepted*. When all changes in the sync queue are accepted, the process models on both levels are *synchronized*.

Thus, we reach a formal definition of synchronization. Two Process Models P_1, P_2 with a correspondence mapping Z, change queues CQ_1, CQ_2 and synchronization queues $SQ_{1,2}$, $SQ_{2,1}$ are synchronized, if

(3) $\forall x \in (A_1 \cup A_2): \exists(a, x) \in R \vee \exists(x, a) \in R$
$\forall c \in CQ_1: c \in SQ_{1,2}, \forall c \in CQ_2: c \in SQ_{2,1}$
$\forall c \in SQ_{1,2} \cup SQ_{2,1}: c$ *is accepted*
$U = \varnothing$.

V. PROTOTYPE

The concept of change management discussed in this paper has been implemented as an extension to Oryx [23], a web-based process modeling framework. We introduced three abstractions levels consisting of subsets of the BPMN 2.0 language constructs. The user establishes relations between process models and activities manually using the *alignmentWizard*. Based on this information, separately implemented change operations manipulate the process

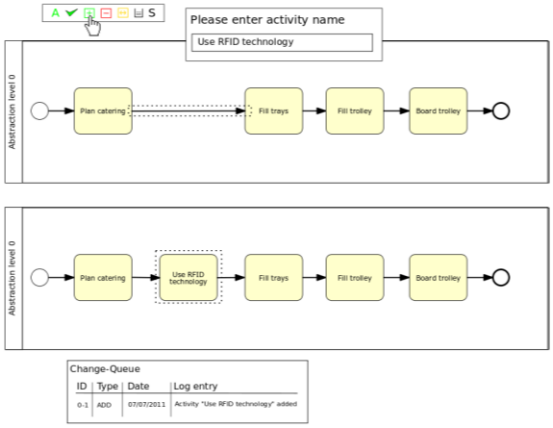

Figure 2. Prototype – Change operation add is performed on abstraction level 0 using a dedicated change operation and synchronization toolbar

model and write to the change queue (Figure 3). At the top, one can see the newly created palette, showing the icons for *alignmentWizard*, check if all elements are aligned, add operation, delete operation, swap operation, change queue, and *synchronizationWizard*. The user marks the arrow and clicks "add". The change operation is stored in the change queue (bottom of Figure 3). The queue is then used to propagate the change to the neighboring level.

Current limitations of the prototype implementation are: (1) Supports only alignment of activities, i.e. no support for gateways or data objects, (2) no automatic alignment procedure as proposed in Section IV.D, and (3) currently, only the subset of change operations shown in Figure 3 is implemented.

VI. Conclusion and Outlook

Managing changes in process models on different abstraction levels is a challenge for which we now proposed a technical solution concept based on model correspondence and change queues for propagating changes. However, managing changes and synchronizing the models is not only a technical issue, but also an organizational one. When changes are made often, the synchronization procedure also has to be performed often in order to limit the amount of changes that have to be considered at a time. Therefore, the applicability of the proposed concept needs to be evaluated in enterprise scenarios. In this work we used BPMN 2.0 on all levels. Although we made use of language sub-setting when we changed a level, we did not perform a synchronization across language boundaries. In principle, our approach can be used for synchronizing graph-based process models using different languages. However, further investigation is required to prove the general applicability of the approach in such scenarios.

Acknowledgment

The work published in this article was partially funded by the openXchange project of the German Federal Ministry of Economy and Technology under the promotional reference 01MQ09011. D. Schumm would like to thank the German Research Foundation (DFG) for financial support within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

[1] C. Wolf and P. Hermon, *The State of Business Process Management 2010*. 2010.

[2] M. Henkel and J. Zdravkovic, "Service-based Processes: Design for Business and Technology," in *Proceedings of the 2nd international conference on Service oriented computing (ICSOC '04)*, 2004, pp. 21-29.

[3] Gartner, "Gartner Reveals Five Business Process Management Predictions for 2010 and Beyond." [Online]. Available: http://www.gartner.com/it/page.jsp?id=1278415. [Accessed: 05-Aug-2011].

[4] M. Weidlich, A. P. Barros, J. Mendling, and M. Weske, "Vertical Alignment of Process Models–How Can We Get There?," *Enterprise, Business-Process and Information Systems Modeling*, pp. 71–84, 2009.

[5] V. Grover, K. D. Fiedler, and J. T. C. Teng, "Exploring the Success of Information Technology Enabled Business Process Reengineering," *IEEE Transactions on Engineering Management*, vol. 41, no. 3, pp. 276-284, 1994.

[6] R. Dijkman, M. Dumas, L. Garcia-Banuelos, and R. Kaarik, "Aligning Business Process Models," in *2009 IEEE International Enterprise Distributed Object Computing Conference*, 2009, pp. 45-53.

[7] S. Buchwald, T. Bauer, and M. Reichert, "Bridging the Gap Between Business Process Models and Service Composition Specifications," *International Handbook on Service Life Cycle Tools and Technologies: Methods, Trends and Advances*, 2011.

[8] G. Decker, "Bridging the Gap Between Business Processes and Existing IT Functionality," *Service-Oriented Applications (WDSOA'05)*, vol. 23819, p. 17, 2005.

[9] S. Brockmans, M. Ehrig, K. Koschmider, A. Oberweis, and R. Studer, "Semantic Alignment of Business Processes," in *Proceedings of the Eighth International Conference on Enterprise Information Systems (ICEIS 2006)*, 2006, pp. 191–196.

[10] J. Küster, C. Gerth, and A. Förster, "Detecting and Resolving Process Model Differences in the Absence of a Change Log," *Business Process Management*, 2008.

[11] J. Vanhatalo, H. Völzer, and J. Koehler, "The Refined Process Structure Tree," *Data & Knowledge Engineering*, vol. 68, no. 9, pp. 793-818, Sep. 2009.

[12] C. Gerth, M. Luckey, J. M. Küster, and G. Engels, "Detection of Semantically Equivalent Fragments for Business Process Model Change Management," in *2010 IEEE International Conference on Services Computing*, 2010, pp. 57-64.

[13] R. Dijkman, "Diagnosing Differences Between Business Process Models," in *Business Process Management*, 2008.

[14] B. V. Dongen and R. Dijkman, "Measuring Similarity Between Business Process Models," *Advanced Information Systems*, 2008.

[15] J. Gao and L. Zhang, "On Measuring Semantic Similarity of Business Process Models," in *2009 International Conference on Interoperability for Enterprise Software and Applications China*, 2009, pp. 289-293.

[16] M. Weidlich, M. Weske, and J. Mendling, "Change Propagation in Process Models using Behavioural Profiles," in *Services Computing, 2009. SCC'09. IEEE International Conference on*, 2009, pp. 33–40.

[17] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change Patterns and Change Support Features - Enhancing Flexibility in Process-aware Information Systems," *Data & Knowledge Engineering*, vol. 66, no. 3, pp. 438–466, 2008.

[18] S. Rinderle-Ma, M. Reichert, and B. Weber, "On the Formal Semantics of Change Patterns in Process-aware Information Systems," *Conceptual Modeling-ER 2008*, 2008.

[19] M. Bauer et al., *Intelligentes Catering mit RFID - Prozesse, Logistik und Integration neuer Technologien im Luftfahrtcatering*. Fraunhofer Verlag, 2010.

[20] Object Management Group/Business Process Management Initiative, "Business Process Model and Notation (BPMN) Version 2.0." [Online]. Available: http://www.omg.org/spec/BPMN/2.0/. [Accessed: 18-Aug-2011].

[21] B. Silver, *BPMN Method and Style: A levels-based methodology for BPM process modeling and improvement using BPMN 2.0*. Cody-Cassidy Press, 2009.

[22] M. Lind and U. Seigerroth, "Multi-Layered Process Modeling for Business and IT Alignment," in *Hawaii International Conference on System Sciences*, 2010, pp. 1–10.

[23] "The Oryx Project," 2010. [Online]. Available: http://bpt.hpi.uni-potsdam.de/Oryx/WebHome. [Accessed: 27-Oct-2010].