Institute of Architecture of Application Systems

Synchronization of Adaptive Process Models Using Levels of Abstraction

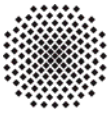
Monika Weidmann[1], Falko Kötter[1], Thomas Renner[1],
David Schumm[2], Frank Leymann[2], Daniel Schleicher[2]

[1] Fraunhofer Institute for Industrial Engineering IAO
and University of Stuttgart IAT
Stuttgart, Germany

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

BibTeX:

```
@inproceedings{Weidmann11,
  author    = {Monika Weidmann and Falko Kötter and Thomas Renner and
               David Schumm and Frank Leymann and Daniel Schleicher},
  title     = {Synchronization of Adaptive Process Models
               Using Levels of Abstraction},
  booktitle = {Proceedings of the 4th International Workshop on
               Evolutionary Business Processes (EVL-BP 2011),
               in conjunction with the 15th IEEE International
               EDOC Conference (EDOC 2011)},
  year      = {2011},
  publisher = {IEEE Computer Society}
}
```

Universität Stuttgart
Germany

Synchronization of Adaptive Process Models Using Levels of Abstraction

Monika Weidmann, Falko Koetter, Thomas Renner
Fraunhofer Institute for Industrial Engineering IAO
and University of Stuttgart IAT
Stuttgart, Germany
Email: firstname.lastname@iao.fraunhofer.de

David Schumm, Frank Leymann, Daniel Schleicher
Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
Email: lastname@iaas.uni-stuttgart.de

***Abstract*—Today many companies use several technologies, modeling languages, and software tools for designing, analyzing, and executing their business processes. The need for adapting processes to new requirements, to reuse parts of processes, and to involve different stakeholders in the process design leads to process changes on multiple process models of different granularity and level of abstraction. These changes cause a need for process models on different abstraction levels to be synchronized in order to avoid inconsistencies. To bridge the resulting Business IT gap, we introduce an approach which supports the creation and adaptation of business processes on different abstraction levels based on reusable process building blocks. The advantage of the approach is that changes of the process can be driven by IT and Business in the same manner, though on different levels of abstraction. In addition to the methodology for this approach, we define reusable process building blocks, describe sychronization mechanisms, and propose a supporting infrastructure. We show the application of these concepts in a real world case study.**

***Keywords*-Adaptive Business Processes, Sychronization, Abstraction Levels, Business Process Management**

I. Introduction

Many companies have recognized that adaptive cross-company business process management is important for their business success [1]. Changing market conditions and legal requirements force companies to work towards adaptive business processes [2]. Additionally, the increasing adoption of business process automation allows providing interfaces for outsourcing process steps and using electronic services from all over the world. However, most companies do not yet execute their business processes in terms of business process technology, but are in early stages of process definition, for example in textual form or with non-executable notations [3] [4]. Most companies see Business Process Management (BPM) as either a top down methodology for process management or a systematic approach for process management [3]. However, the need to involve various stakeholders in process modeling causes that processes are modeled on more than one abstraction level. Combined with changing requirements, this leads to the problem to propagate changes in process models through various abstraction levels. As each level provides a different view on the same process, there is a need for consistency between levels. To satisfy this need, process models on different abstraction levels need to be *synchronized*.

Though some tools promise support for round trip business process management[1], to the best of our knowledge, state-of-the-art tools in production do not support synchronization of template-based adaptive process models on various abstraction levels. Additionally, in research a need has been identified to provide a supporting methodology and model management as well as reuse [5].

Then again, many new technologies, tools, and methods already have been developed in research, which support several aspects. Examples are the introduction of views on processes [6], which allow to add or remove complexity to process models, and the adding of variability to process models resulting in adaptable business processes like in [7]. Additionally standards like BPMN [8] have been developed with the goal to give a notational basis.

We introduce an approach which supports adaptive business process management. We describe our vision of template-based adaptive business processes, integrating concepts which have been developed in previous research. We therefore use process building blocks on various abstraction levels in order to enable reuse and describe the synchrionization of process models on different abstraction levels. Additionally, we outline a methodology describing the involved roles and how these concepts can be applied, including the needed tool support. Further on we introduce a use case and show several aspects of the introduced methodology.

This paper is structured as follows. In Section II the relevant related work in business process management in industry as well as in academia is described. In Section III we describe our vision of synchronization of adaptive business process models on different abstraction levels, including a conceptual overview and the description of solution elements as well as their interrelation. A methodology for using our concept in creating adaptable multi-level business processes is given in Section IV. The use case is provided in Section V. Finally a conclusion and an outlook are provided in Section VI.

[1] www.eclarus.com, www.lombardisoftware.com

II. Related Work

In this section we examine related work in the following categories: business process alignment, views on business processes, and adaptive business processes.

In the field of the alignment of business processes much work has been done already. We have seen how business processes can be aligned using various algorithms (semantic, syntactic, structural matching) [9], sometimes considering different abstraction levels. Change propagation has for example been dealt with in [10]. However, the usage of process fragments as reusable incomplete parts of a process model for change propagation across abstraction levels has not been in the focus. [11] gives an overview of change patterns of business processes. These patterns generally specify which changes of a business process are possible. Examples for such changes are parallelization of activities or the replacement of process fragments. The concept of vertical alignment of process models in general has been reviewed in [12]. The problems of alignment have been stated, but no change propagation method has been proposed. A generic method for handling dependencies across different models has been presented in [13]. The changes to process models during their complete lifecycle have been studied in [14]. To summarize the existing approaches, research has been done on alignment and change propagation in models in general and partly also on process models in detail. There still persists the need to define a structured approach for enabling adaptivity in process models on different abstraction levels, to describe the synchronization of these levels including a methodology with different roles and stakeholders, and to introduce supporting tools.

An important related research topic deals with views on business processes [6] [15]. In contrast to the work in the field on alignment of business processes, views can be used to specify different viewpoints on a single business process model. Considering one model as the main process, views can show certain aspects - like aggregation of information or fading out information. If a change is done in the process model, the view definition is still valid and therefore the view of the process model is updated automatically after the change. The view itself is typically read-only and not used for changing process models itself.

In the fields of creating flexible, adaptive, or configurable business processes various approaches have been introduced [16] [7] [17] [18]. These approaches allow modeling processes at design time with certain flexibility, enabling the creation of several process variants. Additionally, most approaches allow modeling of dependencies between design decisions. However, these approaches typically do not consider abstraction level-based decisions. The synchronization across abstraction levels has not been considered so far using these dependency concepts. In the field of reuse of parts of business processes the notion of process fragments is gaining momentum [19]. However, using these fragments for adapting processes top down or bottom up has not yet been discussed. In the field of compliance an approach for introducing refinement layers and propagation of compliance requirements through business process layers has been introduced [20].

The notion of abstraction levels is inherent to the field of business process design today - they have been mentioned and introduced in [8] and [21] among others. However, the problem of change propagation is not addressed here.

To summarize the existing approaches, none of them have introduced the concept of *adaptive business processes on abstraction levels*, considering the *synchronization* as well as the *reuse of process building blocks*. In this paper, we will extend the approach of Adaptive Business Process Modelling in the Internet of Services (ABIS) [18] in order to support these use cases. In the next section, we will describe our approach for fulfilling these requirements.

III. Abstraction Levels for Adaptive Business Processes

In this section we (1) describe the main concept of our solution, (2) define the notion of abstraction levels in the scope of this paper, and (3) propose an infrastructure for synchronization of process models across different abstraction levels. In contrast to previous approaches, we do not try to solve change propagation in business processes in general, but we use a process template based concept. We describe reusable *process building blocks* and their dependencies across different abstraction levels in order to enable adaptation by various stakeholders. Additionally, we introduce a methodology, showing which stakeholders can use which methods and models in order to design and change process models at various abstraction levels, and describe how these process models are synchronized. We choose BPMN 2.0 as notational basis in this paper, but the approach could be extended for other notations.

We introduce solution elements according to the conceptual model shown in Figure 1. It shows the three solution elements concerning (1) the adaptive process, (2) process synchronization, and (3) the infrastructure. In the course of the next sections these concepts are defined and described in detail.

A. Underlying Concept: Abstraction Level

In context of the conceptual model in Figure 1, the abstraction level (see top of the figure) is the main concept introduced in this paper. A business process is defined in several levels, differing in detail and abstraction. These n *abstraction levels* L_k, where $k = 0..n-1$ form a hierarchy, from the most abstract (top) level L_0, to the least abstract (bottom) level $L_n - 1$. Depending on the modeling task, n has to be chosen carefully. According to existing research

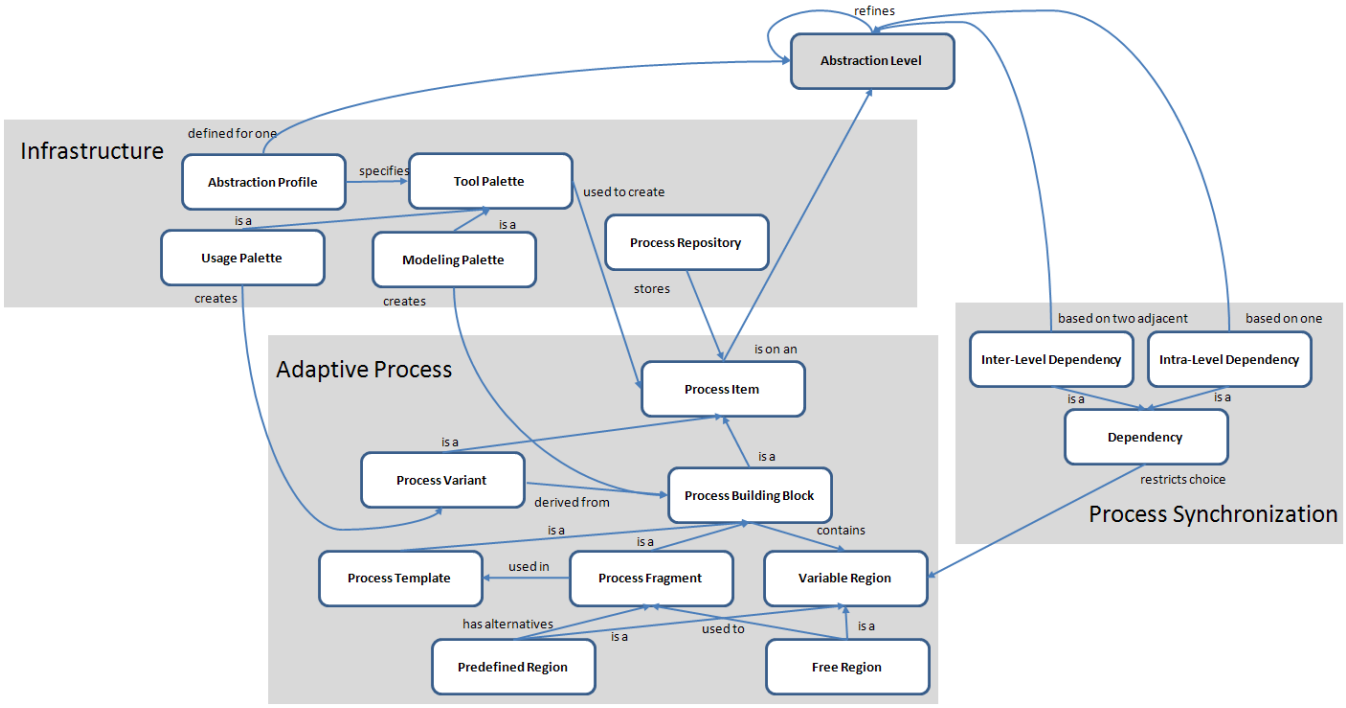


Figure 1. Conceptual model for synchronization of adaptive business processes on different levels on abstraction

and the state of the art [21] [8], we chose $n = 3$ in the scope of this paper:

- **General Business Level** L_0 is the level for business users with basic skills in process design, however not with focus on technical aspects. Only basic modeling capabilities without any technical details are available. This is the most abstract level providing the best overview for the management.
- **Analytical Process Level** L_1 is a more detailed description of the process. It includes advanced elements, which require a deeper understanding of business process semantics. Here, error handling and advanced event handling are added.
- **Detailed Technical Level** L_2 In contrast to the analytical level, this level contains technical details like web service endpoint references, fault handlers, and other process execution semantics. It is used to define an executable business process for a workflow engine.

B. Concept 1: Adaptive Process

In Figure 1 the supporting concepts for adaptive processes are shown in the lower left corner. Depending on the previous work, *process fragments* are usually defined as incomplete processes which cover a reusable part of functionality [19]. In [18] we introduced a notation which allows modeling process fragments in BPMN through the usage of special variability elements. These process fragments can then be inserted into so-called *process templates* to create *process variants*. To achieve this, process templates contain *variable regions*. A variable region is a new BPMN element which serves as a placeholder for a process fragment chosen from a list of *alternatives*. We will refer to process fragments and templates as *process building blocks*, as they are used to enable the creation and adaptation of process variants (see Figure 1). We introduce the notion of *process items*, which are either process building blocks or process variants.

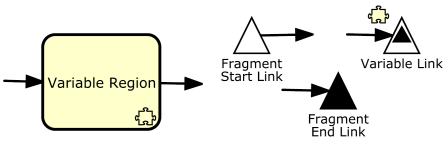


Figure 2. Selected ABIS modeling elements

For the scope of this paper the concepts of a variable region, a fragment start link, a fragment end link, and a variable fragment link are relevant (see Figure 2). Each process fragment contains one *fragment start* and *fragment end link*, which serve to connect an inserted fragment to the incoming and outgoing sequence flows of the corresponding variable region. Process fragments are modeled in BPMN and can contain *variable links*, at which additional sequence flows can be connected to and from the *process template* during insertion. Figure 3 shows an example of such an insertion. The process fragment on the left contains one fragment start link, one fragment end link, and one additional variable link. It is inserted in the process template in the middle. The

result is shown on the right. Note that the fragment links are resolved during insertion.

For the purpose of this paper, we extend the process template with the notion of a *free region* - a free region is a variable region for which no process fragments have been predefined. Thus, a fragment may be modeled on the fly during variant creation.

C. Concept 2: Process Synchronization

In addition to the concepts for adaptive processes, the synchronization of the process models needs to be supported. Elements for this task are shown in the right of Figure 1. We introduced a dependency concept in [22]. It is possible to define dependencies between the choices of variable regions in order to (1) define in which order they are filled and (2) to limit the alternatives at a choice depending on previous choices. For this work, we define that a variable region 1 depends on a variable region 2 if the choice for variable region 2 has to be made first and affects the alternatives for variable region 1. We described the dependency concept only for dependencies in one process model, that means that no abstraction levels have been considered so far. We call these dependencies *intra-level dependencies* in this work.

To support or automate propagation of changes between abstraction levels to synchronize the process models, some kind of matching between elements and selected fragments on different levels has to be introduced. We extend the existing concept to support *inter-level dependencies* from one level to another. Consider a variable region on level L_k, which is neither the highest nor the lowest level. The choice at this variable region may depend on other choices on L_k which have to be made beforehand, as well as on choices made on the levels $L_{0...k-1}$. In turn the choice made at this variable region may influence choices on level $L_{k+1...n-1}$. Therefore, we introduce additional *inter-level dependencies* in between two abstraction levels L_k and L_{k+1}. As the business logic is defined at the uppermost level and should be unaware of the implementation details, we only consider *top down dependencies*, so decisions on a more abstract level affect the lower level design decisions and not vice versa. Thus, a level L_k is influenced only by the levels $L_0...L_{k-1}$ above it. To reduce modeling complexity and preserve level isolation, a level L_k can only have inter-level dependencies to the level L_{k-1} directly above it. Other dependencies arise due to transitivity.

Consider a *process template* on level L_k, which is connected to the corresponding process templates on the lower level L_{k+1}. A template on level L_k can influence multiple templates on L_{k+1}. For example, if on a more detailed level one process is split into one main process and several subprocesses, adaptation of the higher level process influences the main process as well as the subprocesses. Depending on the *process fragment* chosen in a variable region on level L_k, one or several fragments on the lower level L_{k+1} may be chosen automatically. However, even when no explicit modelling of bottom up dependencies is considered, by evaluating top down dependencies these can also be used to propagate changes bottom up to a higher level. Consider a different fragment chosen on L_k, which in turn violates a dependency to L_{k-1}. If validity is to be preserved, a different choice has to be made on L_{k-1}. We consider three types of changes: top down, bottom up, and middle out:

- *Top down*: Considering the creation of a business process, we start from the level L_0 and then create $L_1, L_2, .., L_n$, using the appropriate *process template* of each level. Decisions like the choice of process fragments to be inserted in the respective process template, are propagated top down throughout the process model using *inter-level dependencies*. Additionally, comments can be made on the higher level, specifying in a textual notation the behavior in complex situations, which are not supported by the higher level modeling palette. Therefore, the missing pieces are to be filled in on the lower level and correlated to the comment they implement.
- *Bottom up*: Details are added on a lower level, which fit to one of three cases: (a) the additions correspond to a specific comment on the higher level, (b) they affect the higher level process model, or (c) they do not affect the higher level process model. In case (a) the correlated comment needs to be stored in addition to the diagram information. In case (b) the change needs to be considered and validated by the responsible person on the next higher level. If any dependencies are violated by the low-level-change, an appropriate choice needs to be made on the next higher level. In case (c) no propagation is necessary.
- *Middle out*: Details are changed on a level L_k, where $0 < k < n$, in between two levels L_{k-1} and L_{k+1}. This results in a top down as well as in a bottom up propagation of the affected parts of a process. Bottom up propagation needs to take place first, as the changes on the lower level may lead to irreconcilable constraints on the next higher level, if *inter-level* dependencies cannot be satisfied considering the state of the lower level. If bottom up propagation is successful, top down propagation can take place, which allows the detailed specification of the newly added process information.

Note propagation to another level may lead to recursive propagation from this level onward. For example, a change on the *general business level* L_0 may result in a change of the *analytical process level* L_1 which in turn necessitates a change on the *detailed technical level* L_2. While it is generally possible to create inconsistent process items, the process designer needs to ensure that the dependencies in the process models can be satisfied.

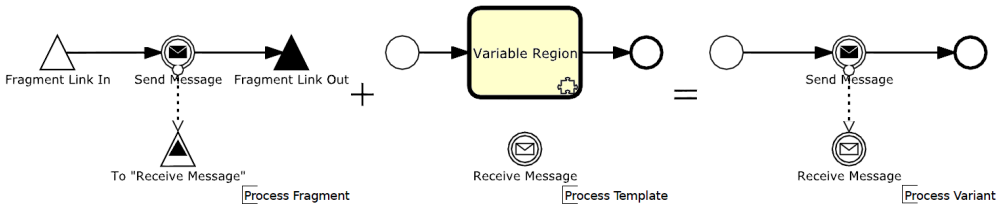


Figure 3. Example for a fragment insertion in ABIS

D. Concept 3: Infrastructure

Here we describe an overview of the resulting solution infrastructure with the following elements:

- *Tool support* for modeling and applying changes to processes
- *Process repository* for reuse and update of process items
- *Tool support for synchronization* of process models top down and bottom up

Tool support For each level L_k a *modeling tool palette* $M(L_k)$ is defined, consisting of a particular subset of BPMN 2.0 elements and the variability modeling elements of ABIS. For the modeling palette $M(L_{k+1})$, the set of elements $M(L_k)$ is extended with the new characteristic elements C_{k+1} : $M(L_{k+1}) = M(L_k) \cup C_{k+1}$. Therefore, on each lower level all elements of the levels above are available. On the lowest level L_{n-1} the complete set of BPMN 2.0 elements is recommended to be supported. For creating and adapting process variants using existing process items, an additional tool palette is needed. We call this a *usage palette*. The definitions of the contents of both tool palettes according to the type of the user are stored in so-called *abstraction profiles*. These match the skill of the user and the right to adapt and model process items to the according tools.

Process repository In order to create process variants from process building blocks, these need to be stored and retrieved accordingly. We consider a process repository to exist, which stores the process building blocks with additional information - which abstraction level it belongs to for example. Additionally some process fragments correspond to other process fragments on other levels. This information needs to be stored in the repository infrastructure as well. A fragment can be positioned on a certain level L_k, if it only contains elements which are allowed on this level.

Tool support for synchronization Considering inter- and intra-level dependencies, automated support for propagating changes or additional process information (like requirements) needs to be introduced for supporting the process modelers. In Table I we show an overview of possible propagation techniques and how these can be used for the top down and bottom up propagation in the use cases of the design of a process and the change of a process model. We came across these solution elements during our case study, where we considered three different change propagation settings as described in Section V.

IV. Methodology

For enabling round trip adaptive business processes at multiple abstraction levels, the adaptive parameters of a business process need to be identified. These can be positioned on various abstraction levels. It is important to distinct higher level design decisions from detailed technical decisions in order to define a *role model* to separate the responsibilities for the abstraction levels. We consider six distinct roles, as we suppose the number of abstraction levels to be three as described in Section III-A and distinguish two types of users: *experts*, who are responsible for modeling *process building blocks* on their respective abstraction levels, and *users*, who create and adapt process variants based on these building blocks. The experts are supported by the modeling tool palette, the business users by the usage tool palette (see Section III-D). This results in a total of six roles: business expert, business user, process expert, process user, execution expert, and execution user.

We consider seven distinct settings. First, we consider the propagation direction of changes: (a) top down, (b) bottom up, and (c) middle out. Additionally, the following use cases are considered: (1) building block creation, (2) variant creation, and (3) variant change.

Top down process building block creation: No process templates exist on any level. A business expert designs adaptive processes creating new process templates and fragments to fill them. On the next lower level, the same template then can be used and extended by the process expert - for example by searching connected fragments on lower abstraction levels, or by explicitly adding or modeling details.

Top down process variant creation: A set of process building blocks for each level exists. A business user creates a process variant using the process building blocks on the highest level, specifying additional requirements by comments. On the next lower level, the appropriate process building blocks are used by the process user. Some choices can be made automatically by evaluating inter-level dependencies. Other choices have to be done manually, either by selecting fragments or by modeling custom elements in a free region, which are then correlated to the additional requirements they satisfy. Likewise, the execution user creates a variant of the template on the lowest level.

Top down process variant change: A business user makes a change on the general business level, either by

Table I
FORMS OF CHANGE PROPAGATION

Support through	Top down propagation	Bottom up propagation	Middle out propagation
Automatic propagation through dependencies	Applicable (possible inconsistencies during change)	-	Applicable (for top down part, which is done last)
Semi-automatic propagation (recommendation) of process fragments	Applicable using dependency concept; possibility of inconsistencies causes need of notification mechanisms	Applicable through extension of dependency concept (going backwards)	Applicable: first bottom up, then top down
Highlighting of propagated changes in the process model for visualization and sign off	Applicable; combine with automatic propagation	Applicable; combine with automatic propagation	Applicable; combine with automatic propagation
Visualization through superimposing possible process fragments on mouse over	Applicable; combine with (semi-) automatic propagation	Eventually applicable (backtracking in the dependency concept to allow bottom up propagation)	Applicable (see top down and bottom up propagation)
Matching of process elements to comments on a higher level	Applicable; show corresponding comments describing requirements during design on lower level	Applicable; Combine with propagation and visualization	Applicable (see top down and bottom up description)
Sign off process for implemented requirements	-	Applicable	Applicable

choosing different fragments or by changing a comment. This change needs to be propagated to the lower levels by the respective designers using inter-level dependencies, as well as choosing fragments manually or by modeling elements in a free region. If new elements are modeled this way, they need to be correlated to the requirement on the upper level.

Bottom up process building block creation: An executable process already exists, albeit it is neither adaptable nor specified on multiple levels. An execution expert takes this process and replaces the parts which shall be variable by variable regions, modeling the extracted parts as process fragments. Alternatively, *automatic pattern recognition* may be used to find existing fragments which are contained in the initial process. Then, additional fragments are defined which enable the creation of differing variants, thus achieving adaptability of the initial process. The resulting process template is then signed off by the process expert, which then adapts the process building blocks for the higher level. After each successive level is created, top down propagation needs to take place so the lower levels satisfy the requirements of the upper levels. Inter-level dependencies need to be added after the next higher levels have been created.

Bottom up process variant creation: A series of process building blocks for each level exist, and a variant shall be created by starting at the lower level. This approach disregards the advantages of the abstraction concept and therefore it is not considered here.

Bottom up process variant change: An execution user makes a change on the detailed technical level. If this change does not affect the higher levels, no propagation is necessary. If the change affects the higher level, it has to be signed off by the process user. If the change passes, it needs to be propagated bottom up. To achieve this, new fragments have to be chosen either manually or by suggestion using inter-level dependencies. Newly modeled elements need to be matched to new or existing elements and/or comments. To assist the user with this task, process building blocks may be superimposed to show the location of lower level changes within the context of the upper level template. The resulting process variant on this level needs to be signed off and propagated as well. If signing off is denied on any level, all changes on lower levels need to be reverted. Finally, if a change is implemented at the highest level, top down propagation needs to take place in case changes on the higher level in turn affect the lower levels.

Middle out process building block and variant creation: These two use cases can be handled as a combination of top down and bottom up approaches, as described above.

Middle out process variant change: A process designer performs a change on the analytical process level. If this change does not affect the higher level, top down propagation can be performed as described. If it affects the higher level, middle out propagation has to take place. First, bottom up propagation is performed as described. Then, if the change is signed off at the top level, top down propagation follows.

V. CASE STUDY

In this section we introduce a real world case study we came across in our work in the openXchange project[2]. We will first describe the basic setting with all levels and process building blocks, before we show how changes are propagated and the process models are synchronized.

A. Basic Setting

We consider a process of active claims management in the insurance industry. We call the claimant or injured party *customer*. After an event of damage has happened, it is

[2]www.openxchange-project.de

possible for the insurance to replace or repair the broken commodity instead of paying money or bills for the customer. Figure 4 shows the process building blocks of this example for the abstraction level L_0, consisting of one process template and five process fragments. Depending on the contract it might be necessary for the insurance to get the consent of the customer - either for each action to be taken (see fragment $0B$), or for all actions at once (see fragment $0C$), or not at all (see fragment $0A$). The action of assignment can be done either by placement of the assignment in a pool (see fragment $0D$), where possible appointees can sign up for assignments, or by directly searching for a matching appointee (see fragment $0E$). In the real world use case other constellations are also possible, but for reasons of space we consider only the setting described above.

As already shown in Figure 2 the ABIS modeling elements are used. Most important are the triangles, which show where a process fragment will be connected to the corresponding process template. The white and black triangle show the connection of the incoming and outgoing edges in the variable region, whereas the semi-filled triangle is a variable link, which allows additional incoming and outgoing sequence flows. It is variable in order to ensure higher reusability in various process templates.

On level L_1 the use case is refined, showing not only the basic steps, but more details about the error handling and the course of events (see Figure 5). Note that additional BPMN elements are used, for example the event-based gateway. The process is now modeled using the loop construct of BPMN, also considering the possibility of a time out. The example contains five variable regions and five process fragments on this level. We do not consider the level L_2 for reasons of space.

An overview of the process building blocks and their interconnection is shown in Figure 6. The process template on level L_0 on the top of the picture contains two variable regions. For variable region 01 the corresponding alternatives are process fragments $0A$, $0B$, and $0C$, for variable region 02 process fragments $0D$ and $0E$ can be used. On the lower half of the figure one can see the process template on level L_1 with a total of five variable regions, one of which is a *free region* - that means for this region no predefined alternatives (process fragments) exist and it needs to be modeled separately.

Within level L_0 no intra-level dependencies exist. However, a total of four inter-level dependencies have been defined. The choices at variable regions 11 and 12 depend on the choice in variable region 01, whereas the choices in variable regions 13 and 14 depend on the choice of variable region 02. On L_1, one can also see that two intra-level dependencies are defined - variable region 12 depends on 11, whereas 14 depends on 13.

B. Configuration

We consider the following configuration on L_0, where the consent from the customer is handled up front and a pool is used: $VR(01) = 0B, VR(02) = 0D$. Additionally, the business user decides to add a comment to $VR(02)$, which describes that in case of an overdue activity there needs to be checked manually what happened to the assignment. See the complete process variant on L_0 in Figure 7. On L_1 this configuration leads automatically via propagation to $VR(11) = 1A, VR(12) = 1B, VR(13) = 1D, VR(14) = 1C$ (see Figure 8). Note that the fragment $1E$ in Figure 5 is modeled on the fly during variant creation by the process user and already applied in Figure 8.

C. Change 1: Top Down Propagation

We consider top down propagation by *automatic propagation* first. For example, on the business level it is decided that the customer is not asked for his consent, because due to a change in the contracts this is not necessary anymore. Therefore the choice of variable region 01 is changed from fragment $0B$ to fragment $0A$, causing in variable region 11 and in variable region 12 that fragment $1B$ has to be used, which means no consent from the customer is needed on L_1 as well. Changes on a higher level are propagated through inter-level dependencies to the next lower level - if intra-level dependencies are violated, more changes need to be performed at that level. In Figure 6 the dependencies are shown. It is also possible to model that several higher-level decisions and several same-level decisions combined affect a lower-level decision. This accounts for variable region 12 - it depends on the choice in variable region 01 as well as the choice in variable region 11.

A different case is the change of higher-level requirements through *manual edits*. If for example an error management requirement is described in a comment on L_0 at the inserted fragment for variable region 02 *Perform actions*, it is then considered as a requirement for the free region 15. Changes in this requirement are then propagated down and need to be visualized at L_1. The edit on L_1 to fulfill such a requirement needs to be propagated back *bottom up* in order to ensure the edit has reached its goal. In our example, the new process fragment $1E$ is created to be inserted in free region 15. This can be done directly within $L1$ during the binding process, but the process fragment is stored in a reusable way in a process repository anyways (see Section III-D).

D. Change 2: Bottom Up Propagation

On the other hand, if for example a change is proposed for variable region 11 at L_1, this change needs to be propagated backward through the inter-level dependencies and *signed off* at L_0. If not, the change cannot be committed. In our example it might be that on the lower level the process user makes a proposal for the improvement of a process by not asking the consumer for consent for each assignment,

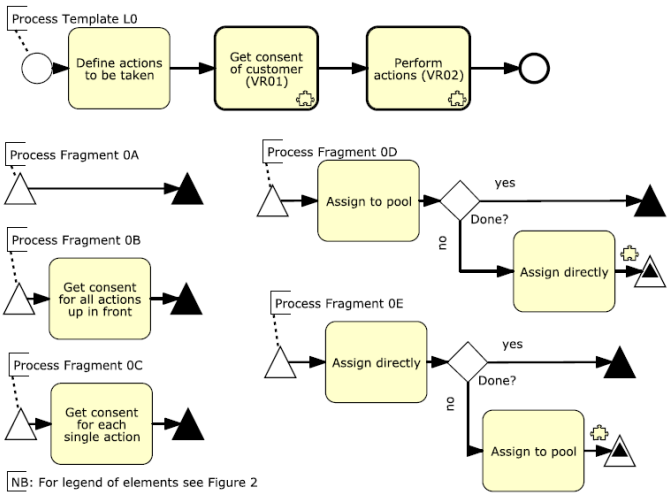


Figure 4. Use case showing process templates and fragments for abstraction level L_0

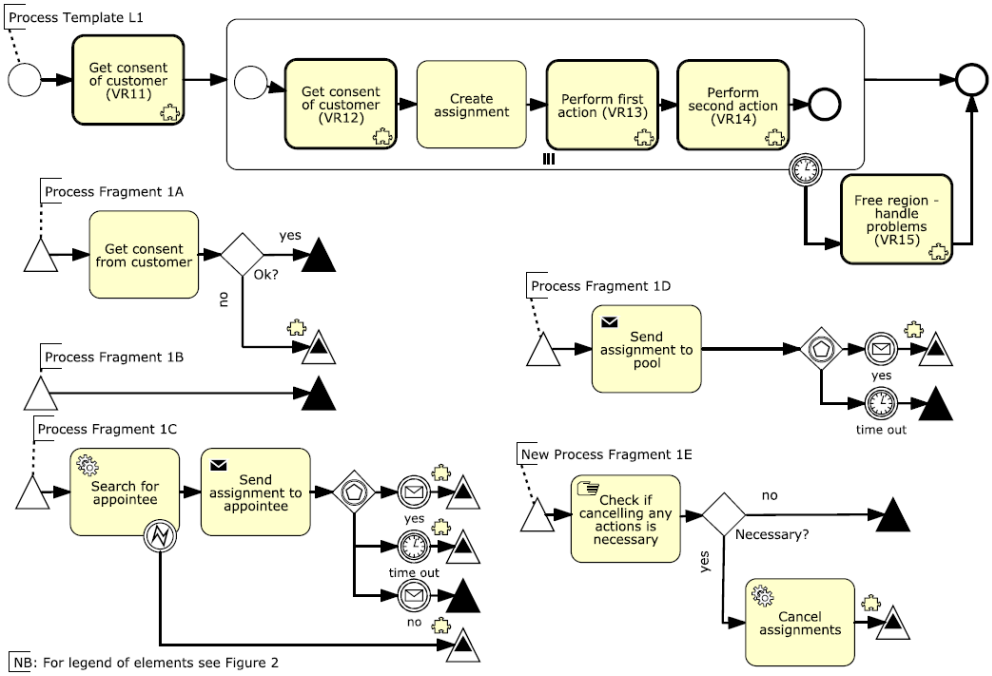


Figure 5. Use case showing process templates and fragments for abstraction level L_1

therefore not using fragment $1A$ at $VR11$, but rather using fragment $1B$ at $VR11$ and $1A$ at $VR12$. Therefore at $VR01$ of the higher level L_0 the corresponding fragment $0B$ needs to be replaced with fragment $0C$, which might be proposed automatically by going backwards through the dependencies and highlighting the change at the higher level. Only after sign off at L_0 the change is finalized.

In case of the free region 15, the process model to fulfill the requirement needs to be signed off at the higher level as well. This could be for example done by showing the process fragment and communicating a description of how the requirement has been fulfilled. This could be approved by a simple confirmation dialog.

In the case of a middle out change on L_1, it needs to be propagated not only to L_0, but also to L_2 (not shown in the example). However, middle out propagation is handled by using mechanisms of bottom up propagation first, *before* the top down mechanisms are applied. The reason for this is that the decisions on the higher levels have usually a higher significance than the ones on the lower levels. If the change is not approved at the higher level, the top down propagation is not performed at all.

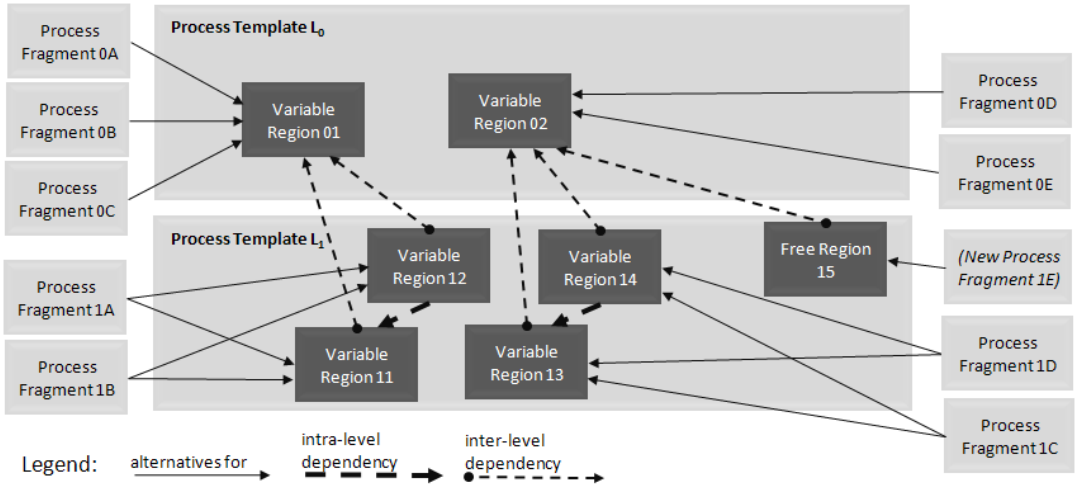


Figure 6. Overview of the use case showing process building blocks and their connections

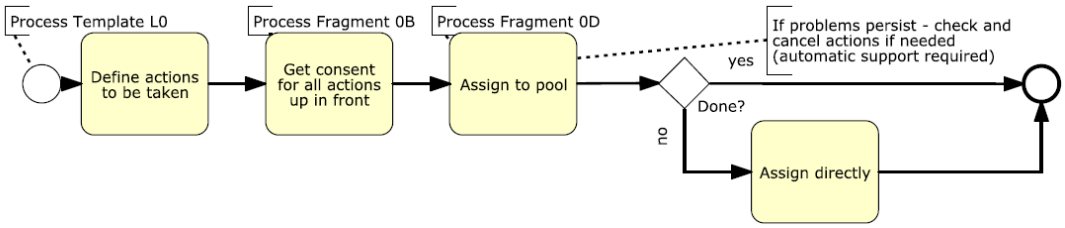


Figure 7. Process variants for L_0

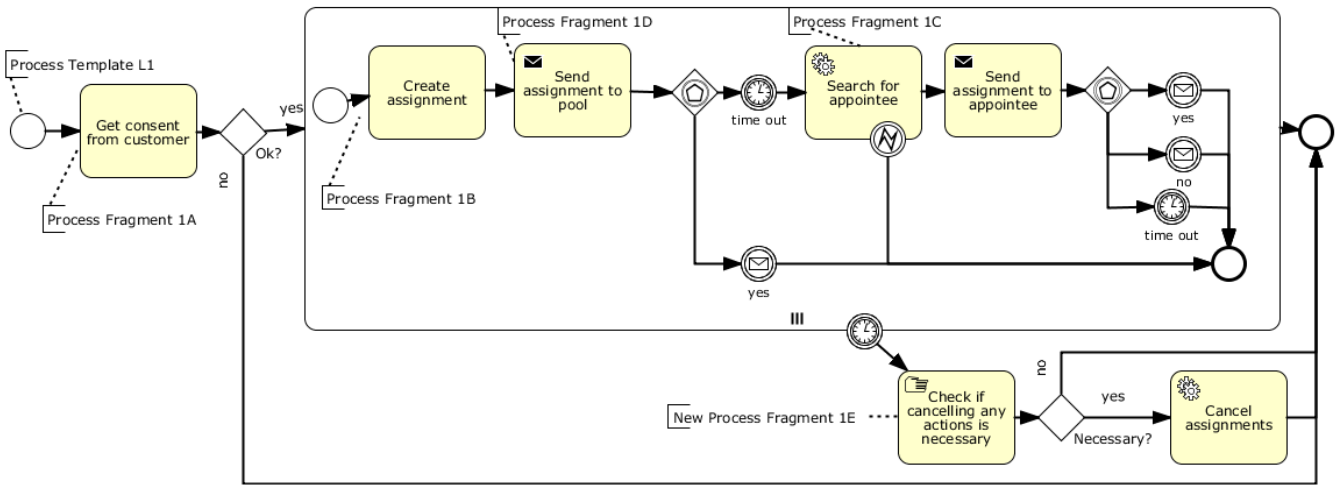


Figure 8. Process variants for L_1

VI. Future Work and Conclusion

In future work we will develop a prototype which extends the currently available concepts of ABIS and apply them to the use case. Further on we will refine the methodology to add more details to the steps in order to enable adaptive business process modeling. We will consider extending the approach in order to combine it with monitoring features and automatically steer the adaptation of processes. Finally we will also consider change propagation in general process models and eventually combine it with the ABIS approach.

In this paper we have introduced abstraction levels. Abstraction levels allow the definition of adaptive process models in different granularity. The adaptivity is ensured by using process fragments in process templates. Changes of process fragments are propagated through the abstraction levels. This allows synchronization of process models on different levels of abstraction. The advantage is that though process models can be standardized, they can still be adapted starting at multiple abstraction levels.

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 and was partially funded by Ericsson. The author D. Schumm would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

[1] N. Dufft, K. Schleife, I. Bertschek, M. Vanberg, T. Böhmann, A. K. Schmitt, and M. Barnreiter, "Das wirtschaftliche Potenzial des Internet der Dienste," p. 199, 2010.

[2] J. Hill, "Five Predictions for How BPM Will Evolve," 2010, last accessed 13.01.2011. [Online]. Available: http://www.documentmedia.com

[3] C. Wolf and P. Hermon, "The State of Business Process Management 2010," *BPTrends Reports (February 2010)*, 2010, last accessed 13.01.2011. [Online]. Available: http://www.bptrends.com/surveys_landing.cfm

[4] S. Patig, V. Casanova-Brito, and B. Voegeli, "IT Requirements of Business Process Management in Practice - an Empirical Study," in *Business Process Management*, ser. Lecture Notes in Computer Science, R. Hull, J. Mendling, and S. Tai, Eds. Springer Berlin / Heidelberg, 2010, vol. 6336, pp. 13–28.

[5] M. Indulska, J. Recker, M. Rosemann, and P. Green, "Business Process Modeling: Current Issues and Future Challenges," in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, P. van Eck, J. Gordijn, and R. Wieringa, Eds. Springer Berlin / Heidelberg, 2009, vol. 5565, pp. 501–514.

[6] R. Eshuis and P. Grefen, "Constructing Customized Process Views," *Data Knowl. Eng.*, vol. 64, pp. 419–438, 2008.

[7] M. Rosemann and W. M. P. V. D. Aalst, "A configurable reference modelling language," *Information Systems*, vol. 32, no. 1, pp. 1–23, 2007.

[8] Object Management Group (OMG), "Business Process Model and Notation (BPMN) Version 2.0," 2009, last accessed 13.01.2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/

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

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

[11] 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.

[12] M. Weidlich, A. 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.

[13] L. Bodenstaff, A. Wombacher, M. Reichert, and R. Wieringa, "MaDe4IC: an Abstract Method for Managing Model Dependencies in Inter-Organizational Cooperations," *Service Oriented Computing and Applications*, vol. 4, no. 3, pp. 203–228, Jun. 2010.

[14] B. Weber, S. Sadiq, and M. Reichert, "Beyond Rigidity - Dynamic Process Lifecycle Support," *Computer Science-Research and Development*, vol. 23, no. 2, pp. 47–65, 2009.

[15] D. Schumm, F. Leymann, and A. Streule, "Process Viewing Patterns," in *Proceedings of the 14th IEEE International EDOC Conference (EDOC 2010)*. IEEE Computer Society Press, Oct. 2010, pp. 1–10.

[16] A. Hallerbach, T. Bauer, and M. Reichert, "Guaranteeing Soundness of Configurable Process Variants in Provop," in *IEEE International Conference on E-Commerce Technology*, 2009, pp. 98–105.

[17] S. Meerkamm, "Configuration of Multi-Perspectives Variants," in *1st International Workshop on Reuse in Business Process Management (rBPM10)*, Hoboken, NJ, USA, 2010.

[18] M. Weidmann, F. Koetter, M. Kintz, D. Schleicher, and R. Mietzner, "Adaptive Business Process Modeling in the Internet of Services (ABIS)," in *Proceedings of the Sixth International Conference on Internet and Web Applications and Services (ICIW)*, 2011.

[19] D. Schumm, D. Karastoyanova, O. Kopp, F. Leymann, M. Sonntag, and S. Strauch, "Process Fragment Libraries for Easier and Faster Development of Process-based Applications," *Journal of Systems Integration*, vol. 2, no. 1, pp. 39–55, 2011.

[20] D. Schleicher, T. Anstett, F. Leymann, and D. Schumm, "Compliant Business Process Design Using Refinement Layers," in *OTM 2010 Conferences*, T. D. et al. R. Meersman, Ed. Springer, Oct. 2010.

[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] F. Koetter, M. Weidmann, and D. Schleicher, "Guaranteeing Soundness of Adaptive Business Processes using ABIS," in *Proceedings of the 14th International Conference on Business Information Systems (BIS 2011)*, Poznan, Poland, 2011.