Markus Nüttgens, Frank J. Rump (Hrsg.)

EPK 2005

Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten

4. Workshop der Gesellschaft für Informatik e.V. (GI)
und Treffen ihres Arbeitkreises „Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)“

08.-09. Dezember 2005 in Hamburg

Proceedings

Veranstalter

veranstaltet vom GI-Arbeitskreis "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)" der GI-Fachgruppe WI-MobIS (FB-WI) in Kopperation mit der GI-Fachgruppe EMISA (FB-DBIS) und der GI-Fachgruppe Petrinetze (FB-GInf).

Prof. Dr. Markus Nüttgens (Sprecher)
Universität Hamburg
Email: markus.nuettgens@wiso.uni-hamburg.de

Prof. Dr. Frank J. Rump (stellv. Sprecher)
FH Oldenburg/Ostfriesland/Wilhelmshaven
Email: rump@informatik-emden.de

Sponsor

SAP AG (Walldorf), Geschäftsstelle Hamburg (Hafencity)

EPK 2005 / Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Hrsg.: Markus Nüttgens, Frank J. Rump. – Hamburg 2005

Vorwort

Ereignisgesteuerte Prozessketten (EPK) haben sich in der Praxis als Beschreibungsmittel für betriebliche Abläufe etabliert. Mit dem Aufbau der Arbeitsgruppe "Formalisierung und Analyse Ereignisgesteuerter Prozessketten (EPK)" im Jahre 1997 wurde ein erster Schritt unternommen, einen organisatorischen Rahmen für Interessenten und Autoren wesentlicher Forschungsarbeiten zu schaffen und regelmäßige Arbeitstreffen durchzuführen (Organisatoren: Markus Nüttgens, Andreas Oberweis, Frank J. Rump). Im Jahr 2002 wurden die Arbeiten der "informellen“ Arbeitsgruppe in den GI-Arbeitskreis "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)" der GI-Fachgruppe WI-MobIS (FB-WI) überführt und inhaltlich erweitert. Der 1. Workshop „EPK 2002“ fand im November 2002 in Trier statt, der 2. Workshop „.EPK 2003“ im Oktober 2003 in Bamberg im Vorfeld der 11. Fachtagung „MobIS 2003“, der 3. Workshop „EPK 2004“ in Luxemburg im Rahmen der GI-Fachtagung „EMISA 2004“.

Der Arbeitskreis soll Praktikern und Wissenschaftlern als Forum zur Kontaktaufnahme, zur Diskussion und zum Informationsaustausch dienen. Die Aktivitäten des Arbeitskreises werden unter der Internetadresse http://www.epk-community.de dokumentiert (aktuell: 150 Mitglieder).

Der vorliegende Tagungsband enthält zehn vom Programmkomitee ausgewählte und auf dem Workshop präsentierte Beiträge. Jeder Beitrag wurde innerhalb der Kategorien Fachbeiträge und Diskussionsbeiträge zweifach begutachtet.

Die Beiträge decken ein breites Spektrum zur Spezifikation und Anwendung Ereignis-gesteuerter Prozessketten (EPK) ab:

- EPK Semantik und Verifikation
- EPK Transformation
- EPK Referenzmodelle
- EPK Metamodellierung
- EPK Sichtenbildung

Der Tagungsband wurde ausschließlich in digitaler Form publiziert und ist frei verfügbar unter: http://www.epk-community.de/epk2005/epk2005-proceedings.pdf.

Wir danken der SAP AG (Walldorf) für die Unterstützung und Bereitstellung der Räumlichkeiten und Verpflegung in der Geschäftsstelle Hamburg (Hafencity) und den AutorInnen und den Mitgliedern des Programmkomitees für die Beiträge zur Realisierung des Workshops.

Hamburg und Emden, im November 2005

Markus Nüttgens
Frank J. Rump

Programmkomitee

Thomas Allweyer, Fachhochschule Kaiserslautern
Jörg Becker, Universität Münster
Jörg Desel, Katholische Universität Eichstätt
Andreas Gadatsch, Fachhochschule Bonn-Rhein-Sieg
Ekkart Kindler, Universität Paderborn
Peter Loos, Institut für Wirtschaftsinformatik / DFKI Saarbrücken
Jan Mendling, Wirtschaftsuniversität Wien
Markus Nüttgens, Universität Hamburg (Vorsitz)
Andreas Oberweis, Technische Universität Karlsruhe
Michael Rebstock, Fachhochschule Darmstadt
Michael Rosemann, Queensland University of Technology
Frank J. Rump, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven
Oliver Thomas, Institut für Wirtschaftsinformatik / DFKI Saarbrücken

Organisation

Markus Nüttgens, Universität Hamburg
Frank J. Rump, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven
Alexander Will, Universität Hamburg

Inhaltsverzeichnis

Fachbeiträge

Diskussionsbeiträge

On the semantics of EPCs: Faster calculation for EPCs with small state spaces

Nicolas Cuntz
Computer Graphics and Multimedia Systems Group, University of Siegen, Germany
nicolas.cuntz@uni-siegen.de

Jörn Freiheit
Max-Planck-Institute Saarbrücken, Germany
freiheit@mpi-sb.mpg.de

Ekkart Kindler
Computer Science Department, University of Paderborn, Germany
kindler@upb.de

Abstract: One of the main features of *Event driven Process Chains* (*EPCs*) is the non-local semantics of the OR-join and the XOR-join connectors. Simulating this non-local semantics faithfully and efficiently is still a challenging problem. A year ago, we have shown that the semantics of moderately sized EPCs can be calculated in reasonable time by using techniques from symbolic model checking. For larger EPCs, however, this method still takes too long for practical use.

In this paper, we introduce and discuss a new technique for calculating the semantics of EPCs: We combine an explicitly forward construction of the transition system with a backward marking algorithm for checking the non-local constraints. Though this method does not always provide a result, it works for most practical examples and, in most cases, it is much faster than the symbolic algorithm. Basically, the computation time is linear in the size of the resulting transition system. The algorithm works for large EPCs as long as the resulting transition systems are small (where transition systems with millions of states and transitions are still considered to be small), which is true for many practical EPCs.

1 Introduction

Event driven Process Chains (*EPCs*) have been introduced in the early 90ties for modelling business processes [KNS92]. For easing the modelling of business processes with EPCs, the informal semantics proposed for the OR-join and the XOR-join connectors are *non-local*. This non-locality results in severe problems when it comes to a formalisation of the semantics of EPCs [LSW98, Rit00, vdADK02]. These problems, however, have been resolved by defining a semantics for an EPC that consists of a pair of two correlated transition relations by using fixed-point theory [Kin04]. Moreover, these transition systems can be calculated by using techniques from symbolic model checking [CK04b, CK05], which is implemented in an open source tool called EPC Tools [CK04a].

For moderately sized EPCs, this tool calculates the semantics in a reasonable time and, therefore, allows us to simulate moderately sized EPCs. For larger EPCs, however, this technique does not work anymore, even if the set of reachable states of the EPC is small. Therefore, we come up with another algorithm for calculating the semantics of EPCs that is tuned to large EPCs with small state spaces. Unfortunately, this algorithm does not provide a result for all EPCs. For some EPCs, the algorithm might fail. But, it works well for most practical examples. And the computation speed is limited only by the available main memory. Our prototype implementation in Java can calculate transition systems with one million states and eight million transitions in 45 seconds on a Linux machine with 1GB main memory and a 2.4 GHz processor. In combination with another optimisation called chain-elimination, which we introduced for improving the symbolic algorithm already, we can simulate EPCs with more than a billion reachable states.

In this paper, we briefly rephrase the syntax and semantics of EPCs (Sect. 2) before presenting the new algorithm for constructing the transition system of an EPC (Sect. 3). Moreover, we will discuss the efficiency of the algorithm and some improvements (Sect. 4). Finally, we will discuss some practical experiences with the new algorithm (Sect. 5).

2 Syntax and Semantics of EPCs

In this section, we informally introduce the syntax and the semantics of EPCs as formalised in [Kin04], which is a formalisation of the informal ideas as presented in [KNS92, NR02].

2.1 Syntax

Figure 1 shows an example of an EPC. It consists of three kinds of *nodes*: *events*, which are graphically represented as hexagons, *functions*, which are represented as rounded boxes, and *connectors*, which are represented as circles. The dashed arcs between the different nodes represent the *control flow*. The two black circles do not belong to the EPC itself; they represent the current *state* of the EPC: A state, basically, assigns a number of *process folders* to each arc of the EPC. Each black circle represents such a process folder at the corresponding arc.

Mathematically, the nodes are represented by three pairwise disjoint sets E, F, and C, which represent the events, functions, and connectors, respectively. We denote the set of all nodes by $N = E \cup F \cup C$. The type of each connector is defined by a mapping $l : C \to \{and, or, xor\}$. The control flow arcs are a subset $A \subseteq N \times N$. In addition, there are some syntactical restrictions on EPCs, which are not discussed here since they are not relevant for our semantical considerations.

A *state* of an EPC assigns zero or one *process folders* to each arc of the EPC. So a state σ can be formalised as a mapping $\sigma : A \to \{0, 1\}$, which is the characteristic function of the set of all arcs that have a process folder in this state. The set of all states of an EPC will be denoted by Σ.

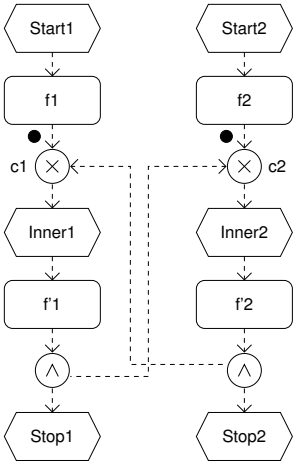

Figure 1: An EPC

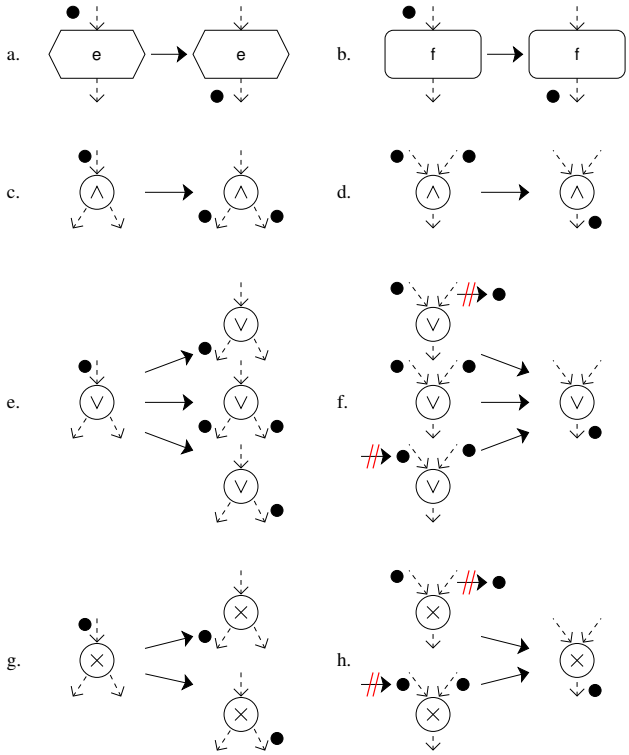

Figure 2: The transition relations for the different nodes

2.2 Semantics

The semantics of an EPC defines how process folders are propagated through the EPC. This can be formalised by a *transition relation* $R \subseteq \Sigma \times N \times \Sigma$, where the first component denotes the *source state*, the last component denotes the *target state*, and the middle component denotes the EPC node that propagates the folders.

For events and functions, the transition relation is very simple: a process folder is propagated from the ingoing arc to the outgoing arc as shown in Fig. 2 a. and b. The semantics of the other nodes is shown in Fig. 2, too. For lack of space, we discuss the details only for the XOR-join connector (case h.): An XOR-join connector waits for a folder on one ingoing arc, which is then propagated to the outgoing arc. But, there is one additional condition: The XOR-join must not propagate the folder if there *is* or there *could arrive* a folder on the other ingoing arc. This additional condition is graphically indicated by the label #•→ at the other arc. Note that this condition cannot be checked locally in the current state: whether a folder can arrive on the other arc depends on the overall behaviour of the EPC. Therefore, we call the semantics of the XOR-join connector *non-local*. The

other node with a non-local semantics is the OR-join connector. Its semantics is shown in Fig. 2 f. Again, the label $\not\bullet$ at an arc indicates the condition that no folder can arrive at that particular arc anymore. The only difference to the XOR-join connector is that an OR-join may fire also when there are folders on both input-arcs.

Note that, in this informal definition of the *transition relation*, we refer to the transition relation itself when we require that no folders can arrive at some arc according to the transition relation. Therefore, we cannot immediately translate this informal definition into a mathematically sound definition. In order to resolve this problem, we assume that some transition relation P is given already, and whenever we refer to the non-local condition, we refer to this transition relation P. Thus, Fig. 2 defines a mapping $R(P)$: for some given transition relation P, it defines the transition relation $R(P)$. Then, the actual semantics of an EPC could be a fixed-point of R, i. e. a transition relation P with $R(P) = P$. Unfortunately, there are EPCs with many different such fixed-points (Fig. 1 shows an example) and there are EPCs that do not have such a fixed-point at all (Fig. 3 shows an example).

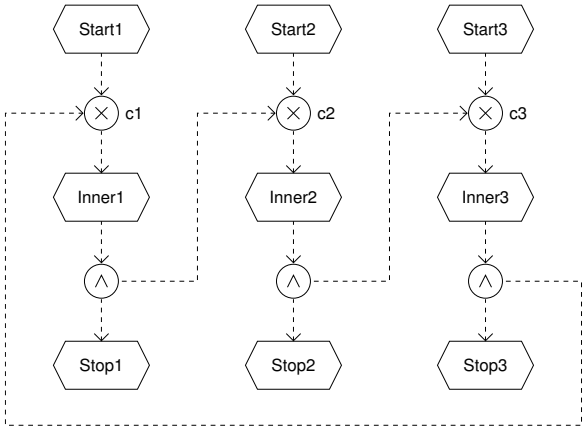

Figure 3: The vicious circle

So, we had to come up with another idea (see [Kin04] for details): The most important property of $R(P)$ is that it is monotonously decreasing in P. This property guarantees that there exists a least transition relation P and a greatest transition relation Q such that $R(Q) = P$ and $R(P) = Q$, where P is called the *pessimistic transition relation* and Q is called the *optimistic transition relation* of the EPC. This pair of transition relations (P, Q) is defined as the semantics of the EPC. In most cases, we have $P = Q$, which means that P is a fixed-point of R. If P and Q are different, there are some ambiguities in the interpretation of the EPC. Therefore, we call an EPC *unclean* if P and Q are different, and we call it *clean* if P and Q are equal.

3 Explicit calculation of the transition system

We have shown that the pair (P, Q) can be calculated by fixed-point-iteration using techniques from symbolic model checking [CK04b, CK05]. Here, we will introduce another algorithm that calculates a transition system P with $R(P) = P$ in an explicit way. In some cases, however, the algorithm might fail.

3.1 Basic idea

The basic idea of the algorithm is very simple: The transition system will be constructed starting from the initial state of the EPC. From this state, we construct new transitions and new states according to the semantics of the different kinds of nodes of the EPC as illustrated in Fig. 2. In the first phase, however, we completely ignore all non-local connectors, i. e. we ignore the OR-join and the XOR-join connectors. In this *local forward construction phase*, we construct all states and transitions reachable by transitions for local nodes only. This algorithm can be implemented in a fairly standard way as known from other modelling notations such as Petri nets.

When no new transitions and states can be constructed in the local forward phase anymore, we start the second phase. In this second phase, we identify those states in which additional folders can be propagated to the input-arcs of the non-local join connectors. To this end, we start a backward marking algorithm for each non-local join connector. Therefore, we call this phase the *backward marking phase*. For each non-local join connector, this phase works as follows: First, we identify those transitions in the already constructed transition system that propagate an additional folder to one of the input-arcs of the non-local join connectors. If a transition adds a folder, we mark the source state of that transition. Then, we systematically mark all the predecessors of these states by going backward through the transition system. When no new states can be marked in this backward marking phase, the second phase stops. Again, the algorithm for the backward marking can be implemented in a fairly standard way.

In the third phase, we investigate all the states of the transition system again and check whether a non-local connector can be fired in one of its states. Since we have marked all states at which additional folders can arrive at the input-arcs of a join connector, it is now easy to decide 'locally' in a state whether a non-local connector can fire or not: If the state is marked by a non-local connector, the non-local connector cannot fire. After adding all transitions (and possibly new states) for the non-local connectors, we begin a new *round* of the algorithm starting with the *local forward construction phase*.

When a round ends without adding new transitions during the third phase, the algorithm terminates. Since the algorithm only adds states and transitions and since there are only finitely many states, we know that this algorithm will eventually terminate.

The question, however, is whether the constructed transition system meets the requirement $R(P) = P$. The answer to this question is quite simple: During the backward marking phase for each non-local connector n, we check for each newly marked state whether

there is a transition for connector n leaving this state already. If this happens during the construction, we know that this transition is wrong, because its non-local condition is violated. As soon as this happens, we know that the constructed transition system violates $R(P) = P$ (we will see an example for such a situation below). If this does never happen, we know that the resulting transition system P meets the condition $R(P) = P$.

3.2 Example

We illustrate the algorithm by constructing the transition system for the simple technical example EPC shown in Fig. 4. We omitted functions from this EPC in order to make the resulting transition system smaller and better understandable. In order to refer to the arcs of the EPC during the construction process, we have named them, and we have also named all nodes including the XOR-join connectors.

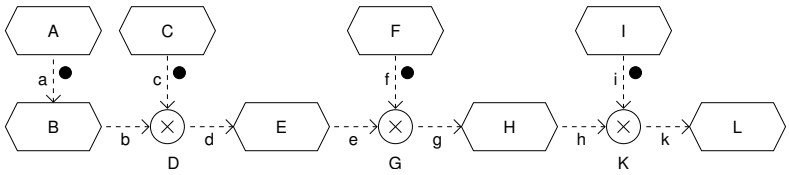

Figure 4: An technical example

We will start the first phase from the initial state of the EPC, which is denoted by $\{a, c, f, i\}$, which is the set of arcs with a folder. Initially, the only local node that can be fired is event B which results in state $\{b, c, f, i\}$. Adding this transition results in the transition system shown in Fig. 5. Note that we cannot add a further transition for a local node to this transition system. So the result of the local forward construction phase is the transition system shown in Fig. 5.

Now, we start the backward marking phase. Clearly, firing event B adds an additional folder to the input-arcs of the XOR-join connector D (viz. the folder at arc b). Therefore, we mark the source state of this transition with D. In order to emphasise its meaning (i. e. connector D should not fire in that state), we represent this mark by a crossed D at the initial node as shown in Fig. 6. Since this node does not have predecessors, the backward marking phase terminates right away – with the result shown in Fig. 6.

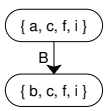

Figure 5: Local forward construction

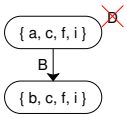

Figure 6: Backward marking

Now, we enter the third phase where we add all possible transitions for the non-local connectors (viz. the XOR-join connectors D, G, and K). The transitions for both connectors G and K can be added to both states as shown in Fig. 7. For the XOR-join connector D, however, we cannot add a transition: The initial state is marked by a D, so we do not add a transition for D in that state; in the second state $\{b, c, f, i\}$, we do not add a transition for connector D because there are folders on both input-arcs, which implies that the XOR-join connector is not enabled in this state. This finishes the third phase of the first round of the algorithm.

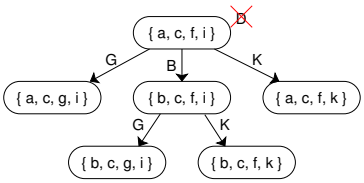

Figure 7: Adding non-local connectors

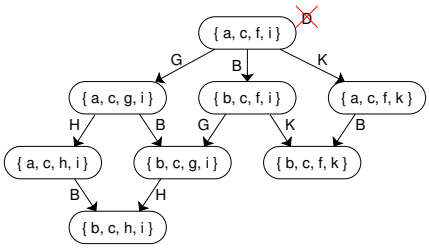

Figure 8: Local forward construction again

Since new transitions have been added during the third phase of the first round, we start another round of the algorithm. We start with the forward construction phase for all local nodes. This gives us the transition system shown in Fig. 8.

Next, we check which transitions propagate an additional folder to the input-arcs of one of the XOR-join connectors. Firing B adds another folder to the input-arcs of XOR-join connector D. So, we mark the source states of the corresponding transitions with a crossed D. Likewise, firing H adds another folder to the input-arcs of XOR-join connector K. So, we mark all the source states of H with a crossed K. The result is shown in Fig. 9. But, the second phase is not finished yet. For all marked states, we must also mark all predecessor states accordingly. The result of this backward marking phase is shown in Fig. 10.

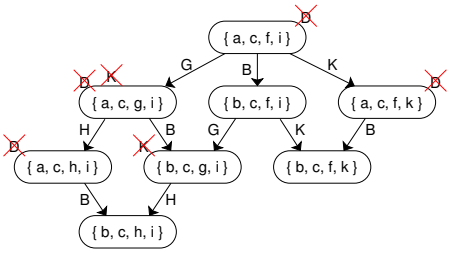

Figure 9: Initialise backward marking

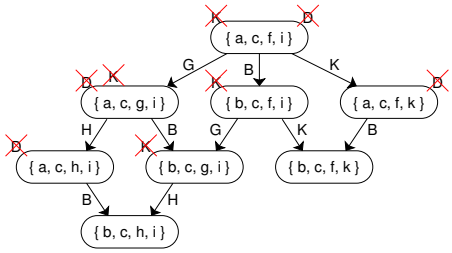

Figure 10: Backward marking

We can see that we have now marked the initial state with a crossed K and we have

constructed a transition K from that state earlier. This shows us that the resulting transition system will not meet the condition $R(P) = P$ and we better had not added transition K to the initial state. This, however, does not bother us right now, we just continue the construction.

We continue our algorithm with the third phase of round two. We add transitions for all enabled non-local connectors. Note that we cannot add transitions for XOR-join connectors D and K since the states are either marked with a crossed D or K, or there are folders on both input-arcs, or there are no folders on the input-arcs at all. We can add a transition only for XOR-join connector G at state $\{b, c, f, k\}$. The result of the second round is shown in Fig. 11.

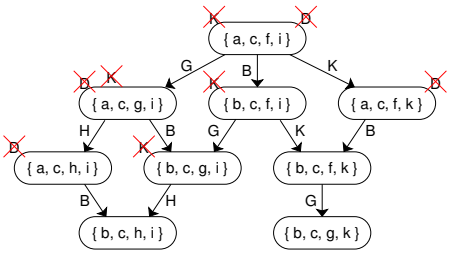

Figure 11: Adding non-local connector

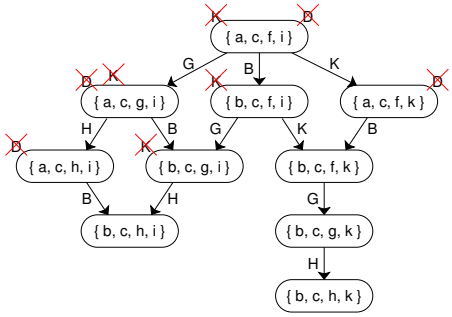

Figure 12: Local forward construction

Since we have added a new transition during the third phase of the second round, we must start another round of our algorithm. In the first phase, we can add only one transition and one state as shown in Fig. 12. Note, that according to the semantics of EPCs, an end event can never fire. Therefore, no transition can be added to the state $\{b, c, h, k\}$.

So, we can proceed with the second phase: We check which transitions add another folder to some XOR-join connector. But, there is none. Note that the new transition adds a folder to the input-arc of XOR-join connector K, but this is the first folder, not an additional folder. So, we do not mark this state.

Next, we check whether we can add another transition for a non-local connector to this transition system. But, this is no longer possible. Note that, in state $\{b, c, h, k\}$, connector K cannot fire due to the folder on its output-arc, which is called a *contact situation*. So, this was the third and last round of our algorithm and the result is shown in Fig. 12. As discussed above, we know that this result is not correct since the transition system has a transition for the XOR-join connector K starting in state $\{a, c, f, i\}$, but this state is also marked with a crossed K now. We will see later that we can improve the algorithm so that this problem is avoided for this EPC.

3.3 Some details

Before discussing this improvement of the algorithm, let us consider some details of the algorithm.

First, let us go into the details of the backward marking algorithm in the second phase. This backward marking algorithm can be implemented in a very standard way. We must make sure only that the data structure for representing the transition system under construction allows us to follow arcs backward. There is only one question left: when or where can we stop the backward algorithm? In our example, we stopped at the initial state. In examples with loops, we would stop as soon as no new states can be marked anymore, which can be implemented in a fairly standard way. However, there is one additional condition when the backward marking algorithm should stop: When the backward marking algorithm for a non-local connector n reaches a transition for connector n, the marking algorithm does not proceed beyond this transition. The reason is that this transition removes all folders from the input-arcs of the XOR-join connector n and, therefore, the preceding state needs not to be marked. This condition can be easily included to the standard marking algorithms since the transitions have labels that refer to the corresponding node of the EPC.

Second, we have seen in our example already, that it is not necessary to start the backward marking in the second phase completely from scratch. We can just keep the markings from the earlier rounds and start from that point. This way, the backward marking algorithm will visit each state of the final transition system at most once. Moreover, it is not necessary to run through the complete set of transitions in order to find those transitions that add another folder to the input-arcs of a non-local connector. We can check this during the creation of a new transition (for each non-local connector) and then add the source state of this transition to a list which will start the backward marking algorithm. Since such a list is needed in the backward marking algorithm anyway, this is no extra effort. This way, the first phase and the second phase of the algorithm are entangled in the implementation. Actually, these phases could be run completely in parallel; but we did not implement it in parallel in order to avoid necessary synchronisations.

Only when the first two phases, local forward construction and backward marking, are finished, the third phase will be started adding currently enabled non-local connectors. Again, it is not necessary to run through all states of the constructed transition system again. During the forward construction, we add all newly created states to a list. Then, we need to check only the states from this list. This way, we need to investigate each state of the final transition system only once for the enabledness of a transition of each non-local connector (the standard forward algorithm guarantees that this is true for all local connectors too).

The last important issue is to avoid the construction of duplicate states. This can be easily achieved by standard hashing techniques, and will, therefore, not be discussed here.

3.4 Complexity

All the above issues are important for increasing the efficiency of the algorithm. It is easy to see that, basically, we have to deal with each state and each transition exactly once during the forward construction. Moreover, the backward marking algorithm visits each state and each transition at most once. Since we have backward marking algorithms for each non-local connector, the time complexity of the construction algorithm is about[1] $O(k \cdot n)$, where n is the size of the constructed transition system (number of states and transitions) and k is the number of non-local connectors of the EPC. Since k is quite small in typical EPCs, the complexity of the algorithm (in time and space) is, basically, linear in the size of the transition system. Practical experience shows, that the algorithm is limited only by the available main memory. For our prototype implementation in Java, we could compute transition systems with one million states and eight million transitions in 45 seconds on a Linux PC with 1 GB main memory and a 2.4 GHz processor. The computation significantly slows down as soon as the main memory is exhausted and parts of the transition system need to be swapped to disk.

The nice feature of this algorithm is that its complexity is independent of the actual size of the EPC. As long as the resulting transition system remains small, the algorithm works fine. Recall that small means in the order of millions of states and transitions. This is in contrast to our symbolic algorithm that computes the complete semantics of the EPC by model checking techniques [CK04b, CK05]. This algorithm might not be able to calculate the transition system – even if the EPC has only a few reachable states. Actually, it was this observation that inspired us to think of another way to calculate the transition system of EPCs with small state spaces (see Sect. 5).

Unfortunately, the algorithm does not always give us a result as we have seen in our example. The algorithm will always terminate, but, sometimes, the resulting transition system P does not meet the condition $R(P) = P$, which is detected during the calculation.

4 Improvements

The algorithm as presented in the previous section provides correct results for many practical EPCs. However, there are some EPCs for which the result is not correct. In this section, we will make some improvements so that the algorithm returns correct results at least for well-designed EPCs – and we believe that the improved version works for all EPCs that are constructed from the four workflow constructs.

In order to explain the idea of the improvement, we have another look at our example from Fig. 4. What went wrong during the construction of the transition system for this EPC was that we added the transition for XOR-join connector K too early to the initial state (see Fig. 7 and 10). Without firing any XOR-join connector at all (first two phases of the first round of the algorithm), no folder can ever reach arc e and arc h. Therefore, the transitions

[1] Actually, there is an additional logarithmic factor for the hashing algorithms and in some worst-case scenarios k might be the number of all connectors.

for both XOR-join connectors G and K were added to the initial state at the same time (see Fig. 7). In this example, however, we can easily see that a folder can be propagated to arc h only after XOR-join connector G has had a chance to fire. So, we had better constructed the transition for XOR-join connector G first, then investigated the states reachable from these states, and then checked for connector K.

4.1 Levels

The basic idea of the improvement is to have different levels of non-local connectors. In the third phase of the algorithm, we check the non-local connectors level by level. If we added a transition for a connector on one level, we do not consider the non-local connectors on all subsequent levels in this round.

For our example from Fig. 4, we chose three levels: D is on the first level, G is on the second level, and K is on the third level. With these levels, the algorithm works as follows: First, we start the forward construction and the backward marking algorithm as in Sect. 3.2. The result is the same as shown in Fig. 6. Then, we check whether a transition for XOR-join connector D (first level) can be added. This is not possible, so we check for XOR-join connector G (second level). The result is shown in Fig. 13. Since, we added a transition for a non-local connector on the second level, we do not check for transitions for connectors on the third level (connector K) in this round. Rather, we start the next round with the transition system from Fig. 13. The forward construction phase will result in the transition system shown in Fig. 14.

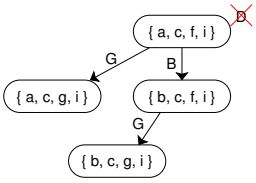

Figure 13: Adding transitions for connector G

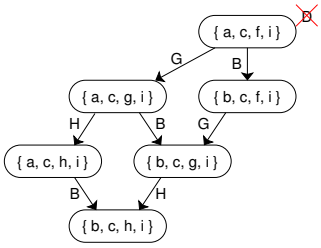

Figure 14: Local forward construction

Next, we start the backward marking phase, which marks the very same states as in our original computation (except for those states that do not occur anymore). The result is shown in Fig. 15.

In the third phase, we check whether transitions for the different levels of non-local connectors can be added. But, it turns out that no non-local transition can be added anymore. So, Fig. 15 shows the final result. This time, there is no transition for a non-local connector starting in a state that is marked. So, the constructed transition system P meets the condition $R(P) = P$, which was called an *ideal* semantics in [Kin04].

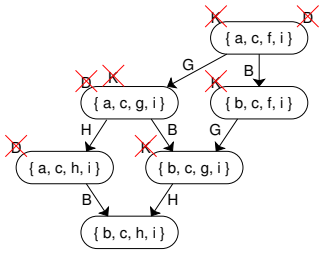

Figure 15: Backward marking algorithm

4.2 Calculating the levels

In our example, we could easily see which non-local connectors should go to which level. In general, however, we need to calculate these levels from the structure of the EPC. Inspired by the above example, we use a simple marking algorithm for the nodes and arcs of the EPC: We start with marking all initial events and set a counter i to 1. Then we proceed with marking the other events and nodes of the EPC as follows (where each node is marked only once):

Events/Functions If an ingoing arc of an event or function is marked, we mark the event or function.

Split connectors If the ingoing arc of split connector is marked, we mark the split connector.

AND-join connectors If all ingoing arcs of an AND-join connector are marked, we mark the AND-join connector.

Arcs If the source node of an arc is marked, we mark the arc.

If no node or arcs can be marked anymore according to the above rules, we proceed as follows:

Fully marked non-local joins First, we check, whether there are unmarked non-local join connectors for which all input-arcs are marked. If this is true, we mark all these connectors and select these connectors for level i. Moreover, we increment the counter i by 1 and start with the first marking phase again

Partially marked non-local joins Otherwise, we check, whether there are unmarked non-local join connectors with at least one marked input-arc. If there is at least one, we mark all these connectors and select these connectors for level i. Moreover, we increment the counter i by 1 and start with the first marking phase again.

Partially marked AND-joins Otherwise, we check, whether there are unmarked AND-join connectors with at least one marked input-arc. If there is one, we mark all these AND-join connectors and start with the first marking phase again.

If no new nodes can be marked anymore, we add all remaining unmarked non-local connectors to the last level and terminate the marking algorithm.

Figure 16 sketches the different phases of the marking algorithm, where the numbers at the different nodes and arcs indicate the level in which the corresponding nodes and arcs were marked. Therefore, the label at each XOR-join connector indicates its level: Connector D is on level 1, connector G is on level 2, and connector K is on level 3.

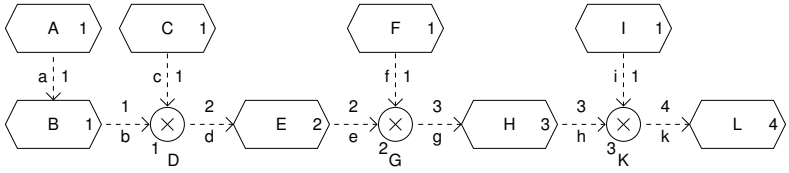

Figure 16: The marking algorithms

4.3 Discussion

The above marking algorithm of an EPC for determining the levels of the different non-local connectors, basically, tries to identify the causal dependencies of the non-local connectors. This order will be considered during the construction of the transition system. The marking algorithm is very efficient; basically, it is linear in the size of the EPC.

Of course, there are EPCs for which the levels calculated by this algorithm do not result in a correct transition system. One example is the vicious circle from Fig. 3; but this is not surprising, because this EPCs does not have a clean semantics at all. We checked our algorithm for several practical examples (e. g. from the SAP/R3 reference models), and it turned out that for almost all examples, our algorithm came up with a correct transition system in virtually no time. There was only one exception, which will be discussed in Sect. 5. Moreover, we believe that for EPCs constructed from properly nested workflow constructs, our algorithm will always result in a correct transition system (a proof of that result, however, is still missing).

5 Practical experience

The quest for another algorithm for simulating EPCs and for calculating their semantics was triggered by some example EPCs from the eJustice project (`www.ejustice.eu.com`), which is funded by the European Union. This project aims at a better understanding

of judicial processes. The discussion started with the EPC shown in Fig. 17 as a screen-shot of EPCTools. This EPC models a German payment order procedure. Here, we do not explain the details of this procedure. Figure 17 should give a rough impression of the structure and size of the EPC model only. The EPC consists of about 90 events, 70 functions, and 40 connectors, where exactly 10 connectors are non-local XOR-joins. One important issue of the eJustice project has been to prove the correctness of the legal procedures originally modelled with ARIS. However, proving correctness requires an adequate simulation tool deriving all reachable states based on a well-defined semantics for EPCs. Although the symbolic algorithm of EPCTools complied with this requirement, it could not cope with the size of the models to be simulated. Therefore, we started thinking of another way of simulating this and similar EPCs.

It was quite obvious that this EPC has only very few reachable states. So, we knew that it should be possible to calculate its semantics somehow explicitly. With our new algorithm, we could calculate the semantics in 70 milliseconds, and it turned out that the transition system had 1215 states and 2551 transitions only.

During tests with practical examples from the SAP/R3 reference models of the ARIS Toolset[2], which were given to us as a benchmark by Jan Mendling, we found only one example for which we could not calculate the semantics. This was the process 'Auftragsabwicklung in der Instandhaltung'; 1GB main memory was not enough to calculate the transition system of this EPC. Therefore, we used a technique proposed in [CK04b] called *chain elimination*, which allows us to reduce the EPC. For this reduced system, we could calculate the semantics. However, the semantics was not clean and there were contact situations, which implied that chain-elimination does not provide correct results. Closer investigation of the model showed that there was an OR-split operation that resulted in quite awkward behaviour of the process. Apparently, this behaviour was not really the intended one. So, we replaced the OR-split by an XOR-split, which appeared to be more adequate. Surprisingly, the resulting EPC could be easily simulated – even without using chain elimination, the transition system could be calculated in 20 milliseconds. This experience shows that long simulation times can give rise to the revision of models, which results not only in better models, but also in faster simulation.

Altogether this shows that, for many practical examples, the new algorithm is a big step forward in the faithful and efficient simulation of EPCs.

6 Conclusion

In this paper, we have presented an new algorithm for efficiently calculating the semantics of an EPC. The idea of this algorithm is explicit forward construction of the transition system combined with a backward marking algorithm, which deals with the non-local conditions.

This idea of a backward marking phase is similar to an algorithm proposed for calculating

[2] ARIS Toolset is a registered trademark of IDS Scheer. For more information see `http://www.ids-scheer.com/`

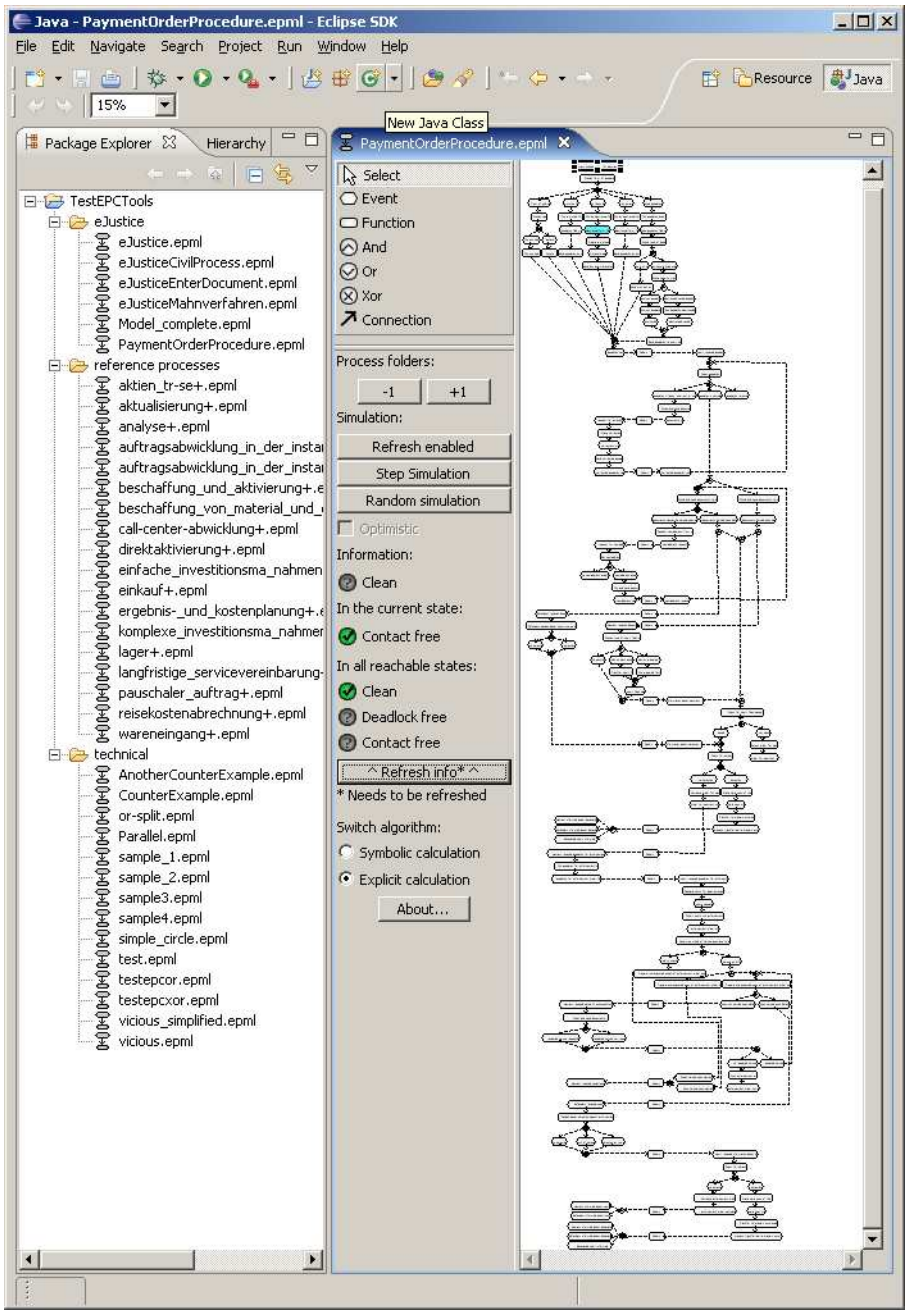

Figure 17: The eJustice example

the enabledness of the OR-join connector in YAWL [WEvdAtH05]; in YAWL, however, only the OR-join connector has a non-local semantics. Moreover, the backward marking algorithm considers only local nodes and the YAWL algorithm does not work in several rounds. Therefore, or algorithm is more adequate for faithfully calculating the semantics of EPCs.

The performance of our algorithm presented in this paper is much better than our symbolic algorithm presented in [CK04b, CK05] – as long as the set of reachable states and transitions are 'small' (i. e. in the magnitude of millions of states and transitions, and, in combination with chain elimination, even billions of states and transitions). The disadvantage of our new algorithm, however, is that in some cases there might be no result at all. Moreover, the new algorithm calculates some ideal transition system. The new algorithm cannot answer the question whether the calculated transition system is the only such transition system or whether there could be more. Anyway, the new algorithm nicely complements the existing algorithm and allows us to simulate (and to analyse) many more EPCs. Experience shows that we could simulate most practical examples based on this algorithm.

The results and measurements come from experiments with a prototype implementation of the new algorithm based on EPCTools. A polished version of this prototype was released in November 2005 as a beta version of EPCTools 2.0. This version supports the explicit and the symbolic simulation algorithm as well as the chain elimination optimisation. It is open source (under the GNU Public License Agreement) and can be downloaded from [CK04a]. What is still missing in this version of EPCTools is functions for analysing the EPC models and for checking some properties. Currently we are investigating and implementing some analysis functions based on the calculated transition system, which will be used for evaluating practical EPCs from the SAP/R3 reference models and from the eJustice project.

References

[CK04a] Nicolas Cuntz and Ekkart Kindler. The EPC Tools Project: Home Page. http://www.upb.de/cs/kindler/research/EPCTools, 2004.

[CK04b] Nicolas Cuntz and Ekkart Kindler. On the semantics of EPCs: Efficient calculation and simulation. In M. Nüttgens and F. J. Rump, editors, *EPK 2004, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, pages 7–26, October 2004.

[CK05] Nicolas Cuntz and Ekkart Kindler. On the semantics of EPCs: Efficient calculation and simulation (Extended Abstract). In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, Second International Conference, 3rd International Conference, BPM 2005, LNCS* 3649, pages 398–403. Springer, September 2005.

[Kin04] Ekkart Kindler. On the semantics of EPCs: Resolving the vicious circle. In J. Desel, B. Pernici, and M. Weske, editors, *Business Process Management, Second International Conference, BPM 2004, LNCS* 3080, pages 82–97. Springer, June 2004.

[KNS92] G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Technical Report Veröf-

fentlichungen des Instituts für Wirtschaftsinformatik (IWi), Heft 89, Universität des Saarlandes, January 1992.

[LSW98] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri Nets 1998, LNCS* 1420, pages 286–305. Springer, 1998.

[NR02] Markus Nüttgens and Frank J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In *PROMISE 2002, Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen, GI Lecture Notes in Informatics* P-21 , pages 64–77. Gesellschaft für Informatik, 2002.

[Rit00] Peter Rittgen. Quo vadis EPK in ARIS? *Wirtschaftsinformatik*, 42:27–35, 2000.

[vdADK02] Wil van der Aalst, Jörg Desel, and Ekkart Kindler. On the semantics of EPCs: A vicious circle. In M. Nüttgens and F. J. Rump, editors, *EPK 2002, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, pages 71–79, November 2002.

[WEvdAtH05] Moe Thandar Wynn, David Edmond, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Achieving a General, Formal and Decidable Approach to the OR-Join in Workflow Using Reset Nets. In G. Ciardo and P. Darondeau, editors, *Applications and Theory of Petri Nets 2005, 26^{th} International Conference, LNCS* 3536, pages 423–443. Springer, June 2005.

EPC Verification in the ARIS for MySAP reference model database

B.F. van Dongen M.H. Jansen-Vullers

Department of Technology Management
Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{b.f.v.dongen,m.h.jansen-vullers}@tue.nl

Abstract: Process aware information systems, such as Workflow Management Systems or ERP systems require process specifications for their implementation. Since many of those process specifications are similar for different companies, databases with so-called reference models, such as Aris for My-SAP, have been developed. These databases can be customized towards company-specific process specifications, thus keeping implementation costs down. To avoid costly problems with information systems on an operational level, it is of the utmost importance that the reference models used to design the information system are correct.

In this paper, we analyze a selection of the reference models for SAP R/3 that are stored in the ARIS for MySAP database, and we verify whether they are correct. Since these models are stored as Event-driven Process Chains (EPCs), we use a verification approach tailored towards the verification of this language to check for errors in the models. We show that the reference models in ARIS for MySAP indeed contain some errors and we present the implications of those errors, if these models would be used for the execution of business processes.

1 Introduction

Nowadays, more and more enterprise information systems rely on process models for their specification. Companies offering large information systems employ consultants who, together with the customer, make process specifications in terms of executable process models. These process models are then used to configure the final installation of the information system. Many of these *process aware information systems* (for example Enterprise Resource Planning (ERP) [KT98] systems and Workflow Management (WFM) [AH02, LR99]) are designed to support business processes on an operational level and to fully benefit from these systems, process models need to be specified as precise as possible. However, designing process models precisely is a complicated and error prone task. Fortunately, process models that are used in different companies, but with similar purposes, often have very similar designs. For this reason, databases have been developed that contain generic process models that can be customized towards company-specific business processes. These process models can serve as a guideline when designing process models to implement large information systems, i.e. they serve as a reference for the designer,

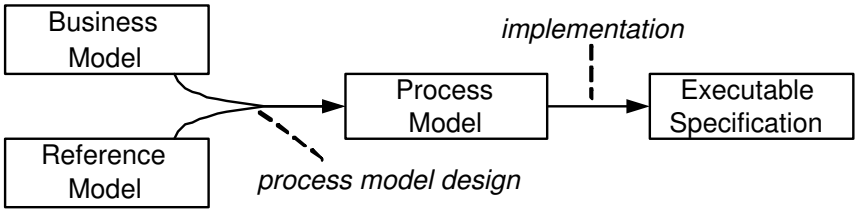

Figure 1: Phases in the configuration of a PAIS

hence the term *reference models*.

For a large information system to support one process, the configuration can roughly be divided in two phases as shown in Figure 1. In the first phase, a reference model that best suits the process under consideration is selected (for example for a purchasing process, this should be the best purchasing reference model, such as vendor selection or internal procurement). Together with the business model of the specific organization, the reference model is customized, resulting in a process model. This process model is used in the implementation phase to configure a specific information system, such as SAP R/3. Although all the steps presented here are usually executed by trained professionals, still they are performed by humans, thus errors are likely to be introduced.

Using reference models in the process model design phase helps to avoid making too many errors. It does however not eliminate the possibility of errors being introduced completely. When using reference models, errors are likely to be introduced for two main reasons. First, errors that are already present in the reference models are likely to be copied into the final process model. Second, process models are designed for each process independently, while in real life, processes are mutually dependent. Any error in a process model used for the implementation phase has a high possibility of leading to severe operational consequences, once the information system is fully implemented. Since the costs of correcting operational errors by far exceeds the costs of correcting modelling errors, it is of the utmost importance that the reference models used are correct.

To find errors in process models, many authors have developed verification methods. Basically, all of these verification methods can be used to check whether a process model is *correct*, in other words, they can be used to check for *correctness* of a process model. In Section 2, we categorize verification methods and we show that some methods look on the level of the executable specification, some on level of the business model and some on process models or reference models.

In this paper, we focus on the correctness of reference models for a specific information system, SAP R/3. The reference models are available in the ARIS for MySAP database in the ARIS Toolset, a commercial product of IDS-Scheer. As a modelling language, the ARIS Toolset uses *Event-driven Process Chains* (EPCs) [KNS92, KT98, Sch94]. We selected SAP R/3, since EPCs are used in a large variety of systems, including SAP R/3. Moreover, SAP R/3 is market leader in the field of Enterprise Resource Planning systems.

Many verification approaches exist for EPCs. The verification method we chose looks at

verification from a designers point of view and assumes the process designer to know what he intends to model. In the field of software engineering, this could be seen as validation, i.e. does the software (in our case the model) do what it is intended to do (in our case, what the process designer intended to model). However, a process specification such as an EPC should not be seen as an executable specification. Instead, the EPC should be seen as the specification of the process. Using this specification an operational system can later be implemented in a company. The approach presented in this paper, verifies the process specification (the EPC) against the possible future behaviour of an operational system. Hence the term verification.

We take the SAP reference models as a starting point, and use the verification approach presented in [DVA05], as our verification method. We show that many of the SAP reference models are correct and can indeed be used without any problems. However, we also show that some of the models should be used with care, i.e., the environment in which they are used needs to satisfy certain conditions for them to be correct. Furthermore, we show that a small number of the reference models is structurally incorrect, i.e., they need to be revised before they can be used as executable models. With respect to these errors, we investigate some common causes, and show how designers could avoid these errors.

The remainder of this paper is structured as follows. In Section 2 we discuss related work with respect to the verification of process models. In Section 3 we describe our domain of analysis: SAP R/3, the EPC modelling method and the reference models. Next, in Section 4, we describe the approach for the verification of these models as implemented in the ProM framework[1], and described in [DMV$^+$, DVA05]. Following this approach we are able to evaluate the SAP reference models in Section 5. This evaluation is based on two lines: the evaluation of one complete module (Section 5.1) and a guided search through the database with reference models (Section 5.2). Finally, in Section 6, we draw some conclusions.

A subset of the work presented here has been published at the International Conference on Business Process Management (BPM05) [DJV05]. That paper however deals mainly with the question if the verification approach of [DVA05] could be used for the verification of reference models. This paper extends [DJV05] by presenting the verification results for the whole module from Section 5.1. Furthermore, we present a guided search trough the reference model database in Section 5.2, illustrating some of the errors related to the independent modelling of dependent business processes.

2 Related work

Since the mid-nineties, a lot of work has been done on the verification of process models, and in particular workflow models. In 1996, Sadiq and Orlowska [SO96] were among the first to point out that modeling a business process (or workflow) can lead to problems like livelock and deadlock. In their paper, they present a way to overcome syntactical errors, but they ignore the semantical errors. Nowadays, most work that is conducted focusses on

[1] See www.processmining.org for details.

semantical issues, i.e. "will the process specified always terminate" and similar questions. The work that has been conducted on verification in the last decade can roughly be put into three main categories, namely "verification of models with formal semantics", "verification of informal models" and "verification by design". In this section, we present these categories and give relevant literature for each of them.

2.1 Verification of models with formal semantics

In the first category we consider the work that has been done on the verification of modeling languages with formal semantics. One of the most prominent examples of such a language are Petri nets [DE95, Mur89, RR98]. Since Petri nets have a formal mathematical definition, they lend themselves to great extent for formal verification methods. Especially in the field of *workflow management*, Petri nets have proven to be a solid theoretical foundation for the specification of processes. This, however, led to the need of verification techniques, tailored towards Petri nets that represent workflows. In the work of Van der Aalst and many others [Aal00, AH00, DR01, HSV03, VA00] these techniques are used extensively for verification of different classes of workflow definitions. However, the result is the same for all approaches. *Given a process definition, the verification tool provides an answer in terms of "correct" or "incorrect"*. However, not all modeling languages have a formal semantics. On the contrary, the most widely used modeling techniques, such as UML and EPCs are merely an informal representation of a process. These modeling techniques therefore require a different approach to verification.

2.2 Verification of informal models

Modeling processes in a real-life situation is often done in a less formal language. People tend to understand informal models easily, and even if models are not executable, they can help a great deal when discussing process definitions. However, at some point in time, these models usually have to be translated into a specification that can be executed by an information system. This translation is usually done by computer scientists, which explains the fact that researchers in that area have been trying to formalize informal models for many years now. Especially in the field of workflow management, a lot of work has been done on translating informal models to Petri nets. Many people have worked on the translation of EPCs to Petri nets, cf., [Aal99, ADK02, DA04, LSW98]. The basic idea of these authors however is the same: "Restrict the class of EPCs to a subclass for which we can generate a sound Petri net". As a result, the ideas are appealing from a scientific point of view, but not useful from a practical point of view.

Also non-Petri-net based approaches have been proposed for the verification of informal modeling languages. One of these ideas is *graph reduction*. Since most modeling languages are graph-based, it seems a good idea to reduce the complexity of the verification problem by looking at a reduced problem, in such a way that correctness is not violated

by the reduction, i.e. if a model is not correct before the reduction, it will not be correct after the reduction and if the model is correct before the reduction, it will be correct after the reduction. From the discussion on graph reduction techniques started by Sadiq and Orlowska in 1999 [SO99, SO00] and followed up by many authors including Van der Aalst et al. in [AHV02] and Lin et al in [LZLC02], it becomes clear that again the modeling language is restricted to fit the verification process. In general this means that the more advanced routing constructs cannot be verified, while these constructs are what makes informal models easy to use.

The tendency to capture informal elements by using smarter semantics is reflected by recent papers, cf. [ADK02, DA04, Kin04]. In these papers, the problem is looked at from a different perspective. Instead of defining subclasses of models to fit verification algorithms, the authors try to give a formal semantics to an informal modeling language. Even though all these authors have different approaches, the goal in every case is similar: *try to give a formal executable semantics for an informal model*.

2.3 Verification by design

The last category of verification methods is somewhat of a by-stander. Instead of doing verification of a model given in a specific language, it is also possible to give a language in such a way that the result is always correct. An example of such a modelling language is IBM MQSeries Workflow [LR99]. This language uses a specific structure for modelling, which will always lead to a correct and executable specification. However, modelling processes using this language requires advanced technical skills and the resulting model is usually far from intuitive.

In this section, we have presented an overview of the literature on process model verification. We have categorized the various methods in three main categories and pointed out why many of them are not used in practice. In this paper, we use the technique presented in [DVA05] that can be seen as a combination of the first two categories. It assumes *the designer* to be able to decide whether or not a specification is semantically correct. This technique has been implemented in the Process Mining (ProM) Framework[2], that is able to import EPCs defined in the ARIS Toolset[3] and provides the designer with feedback about possible problems. SAP reference models are available in the ARIS Toolset format, and the users of these reference models are typically consultants that have a deep knowledge about the process under consideration. Hence, we found the approach described in [DVA05] to be the best approach for the verification of the SAP R/3 reference models.

[2] See www.processmining.org for details.

[3] See www.ids-scheer.com for information about the ARIS toolset.

3 SAP R/3 Reference models

Several authors researched the area of reference models before, see e.g. [Ber, CK97, FL03, Ros03, RA03, Sch00, Sil01a, Sil01b, Fra99]. In this section we introduce reference models based on [RA03] and then explain Event-driven Process Chains (EPCs).

3.1 Reference models

Reference models are generic conceptual models that formalize recommended practices for a certain domain [FL03, Fra99]. Reference models accelerate the modelling process by providing a repository of potentially relevant business processes and structures. With the increased popularity of business modelling, a wide and quite heterogenous range of purposes can motivate the use of a reference model. These purposes include software development, software selection, configuration of Enterprise Systems, workflow management, documentation and improvement of business processes, education, user training, auditing, certification, benchmarking, and knowledge management [RA03].

What we learn from previous authors is that we can distinguish two types of reference models: industry models and application models. Industry reference models are generally higher level models and they aim to streamline the design of enterprise-individual (particular) models by providing a generic solution. Application reference models describe the structure and functionality of business applications including Enterprise Systems. In these cases, a reference model can be interpreted as a structured semi-formal description of a particular application. This application can then be seen as an existing off-the-shelf-solution that supports the functionality and structure described in the reference model.

Rosemann and van der Aalst explain in [RA03] that application reference models tend to be more complex than industry reference models. They explain that the SAP reference model is one of the most comprehensive models [CK97]. 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. In the early nineties, two companies called SAP and IDS Scheer, have developed an intuitive process modelling langauge, which resulted in the process modelling language Event-driven Process Chains (EPCs). This language has been used for the design of the reference process models in the ARIS for MySAP database that we consider in this paper. EPCs also became the core modelling language in the Architecture of Integrated Information Systems (ARIS) [Sch00, KNS92].

3.2 Event-driven Process Chains (EPCs)

The SAP R/3 reference models are modelled as Event-driven Process Chains, or EPCs, in the ARIS Toolset. An EPC consists of three main elements. Combined, these elements define the flow of a business process as a chain of events. The elements used are:

Functions, which are the basic building blocks. A function corresponds to an activity (task, process step) which needs to be executed. A function is drawn as a box with rounded corners.

Events, which describe the situation before and/or after a function is executed. Functions are linked by events. An event may correspond to the position of one function and act as a precondition of another function. Events are drawn as hexagons.

Connectors, which can be used to connect functions and events. This way, the flow of control is specified. There are three types of connectors: $\wedge$ (and), $\times$ (xor) and $\vee$ (or). Connectors are drawn as circles, showing the type in the center of the circle.

Functions, events and connectors can be connected with edges in such a way that (i) events have at most one incoming edge and at most one outgoing edge, but at least one incident edge (i.e. an incoming or an outgoing edge), (ii) functions have precisely one incoming edge and precisely one outgoing edge, (iii) connectors have either one incoming edge and multiple outgoing edges, or multiple incoming edges and one outgoing edge, and (iv) in every path, functions and events alternate (no two functions are connected and no two events are connected, not even when there are connectors in between).

In the ARIS for MySAP reference databases, there are hundreds of EPCs that can be used in many different situations, from "asset accounting" to "procurement" and "treasury". Since we cannot discuss all these models here, we focus on one of the modules that can be considered to be a representative subset of all reference models, namely "procurement". This is a set of some 40 EPCs, all in the area of procurement. They describe processes for (i) internal procurement, (ii) pipeline processing (iii) procurement of materials and external services, (iv) procurement on a consignment basis, (v) procurement via subcontracting, (vi) return deliveries, and (vii) source administration.

All 40 models were analyzed using the approach described in [DVA05]. Before we show the results of this verification process in Section 5, we first briefly introduce this verification approach in Section 4.

4 Verification approach

For the verification of the EPCs in our reference model database, we use the approach described in [DVA05]. This verification approach is tailored towards the verification of Event-driven Process Chains and it assumes *the designer* of an EPC to be able to decide whether or not the EPC is correct. The approach is implemented in the ProM framework ([DMV$^+$]) and it is freely available for download.

The verification process described in [DVA05] consists of several steps. In the first step, the designer of the EPC has to provide the tool with all combinations of initial events that could initiate the modelled process. Using this, the tool calculates all the possible outcomes of the process (in terms of events that occurred and have not been dealt with). Then, the tool requires the designer to divide those outcomes in two groups, the first of which contains all the outcomes that represent the desired behavior of the process. The

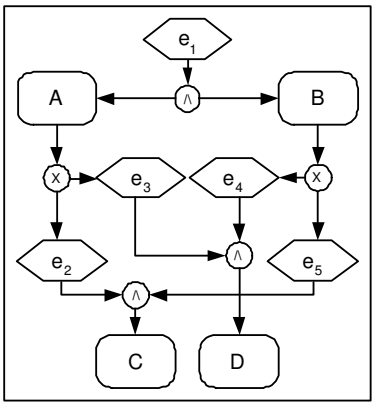

Figure 2: EPC with choice synchronization

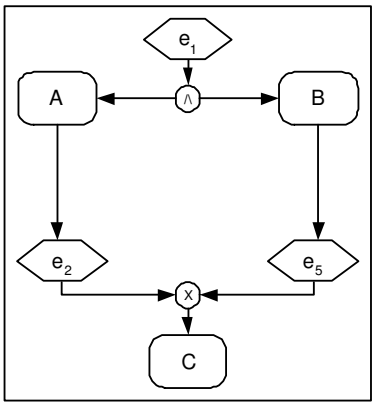

Figure 3: EPC with erroneous routing

second group contains the undesired behavior. Clearly, depending on the model, either of the two groups can be empty.

4.1 Semantically correct models

Models that are semantically correct are models of processes that, when started in any allowed state, will *always* terminate in one of the allowed termination states. In other words, routing constructs do not have to be synchronized. Choices can be made locally, without any knowledge of the execution history.

4.2 Syntactically correct models

Models that are syntactically correct are models of processes that, when started in any allowed state, will always *have the possibility to* terminate in one of the allowed termination states. In other words, routing constructs have to be synchronized. Not all choices can be made locally, instead, the execution history limits the available options. An example of such a construct can be found in Figure 2, where the choices after functions A and B have to be synchronized in order to allow function C or D to execute. However, at any point in time, there is always an option to complete in a correct way. For example enabling event e_3 after doing function A requires that after function B event e_4 is enabled. This can easily be enforced by an operational system.

4.3 Incorrect models

The final class of models are the incorrect ones. These models contain syntactical errors, such as an AND-split followed by an XOR-join or the other way around. An example of such an incorrect model is shown in Figure 3, where functions A and B originate from an AND-split, and are later joined by an XOR-join. As a result, function C will be carried out twice based on the same case in event e_1.

5 Verification of the reference models

The application of the verification approach presented in Section 4 is based on a basic assumption: It assumes that the designer of a model has a good understanding of the actual business process that was modelled, and he knows which combinations of events may actually initiate the process in real life. Typically, reference models are used by consultants that do indeed have a good understanding of the process under consideration. Besides, they know under what circumstances processes can start, and which outcomes of the execution are desired and which aren't. Therefore, the approach seems to be well suited for the verification of the SAP reference models.

5.1 Procurement module

As stated in Section 3 we focus on the procurement module of the ARIS for MySAP reference model database, since it can be seen as a representative subset of all reference models. The procurement module contains several sub-modules and we analyzed all the models from these modules using the approach presented in Section 4. Surprisingly, already in the first model (Internal Procurement) there were structural errors. In Figure 4, we show a screenshot of the verification tool used. It shows part of an EPC in which an AND-split is later joined by an XOR join. Recall Figure 3, where we have shown that this is clearly incorrectly modelled. As a result, if this model would *not* be repaired, payments could be made for goods that were never received. Obviously, this is not desirable. In Figure 5 we show the repaired model, i.e. the XOR-join has been changed into an AND-join. Now, the model is semantically correct, which means that it can be used in a business environment without problems.

The results of our analysis of the whole procurement module are presented in Table 1, which contains three columns. The first column shows the name of the module. The second contains the verification result. We use "I" for incorrect models, "S" for syntactically correct models, and "C" for semantically correct ones. The final column gives the business-wise implication of the error found if this model would be translated into an executable specification, if applicable.

5.2 Guided model selection

From the previous section it seems that we can conclude that most errors are made in the higher level models. Using this as a guide, we tried to find problems in the reference models. In fact, in the high level models, it is not hard to find these mistakes, by manually browsing through the Aris for MySAP Reference model databases. These high level models are usually more complex then the lower level models (i.e. they contain more functions, events and connectors). Therefore, errors are more likely to be introduced there. We would like to mention two observations that we made during this guided model selection.

The first observation is that often, one particular initial event is applied in several (sub-)models. Take, for example, the event "Deliveries need to be planned". This event occurs in 15 different models. Every time it occurs, it is joined with the event "delivery is relevant for shipment". However, in some models this is done via an XOR-join, and in some models via an AND-join. In Figure 6, we show these two events, used in the "Consignment Processing" module, where they are joined by an XOR-join. However, in Figure 7, we show the same two events in an AND-join configuration. Since these two events are always followed by something that refers to transportation, it seems that they should always appear in an AND-join configuration. However, only a designer with deep knowledge of the process that is modelled can decide if that is the case.

The second observation, that seems to be a common one, is the effect of re-use. Typically, many different organizations have very similar processes. Therefore, when building refer-

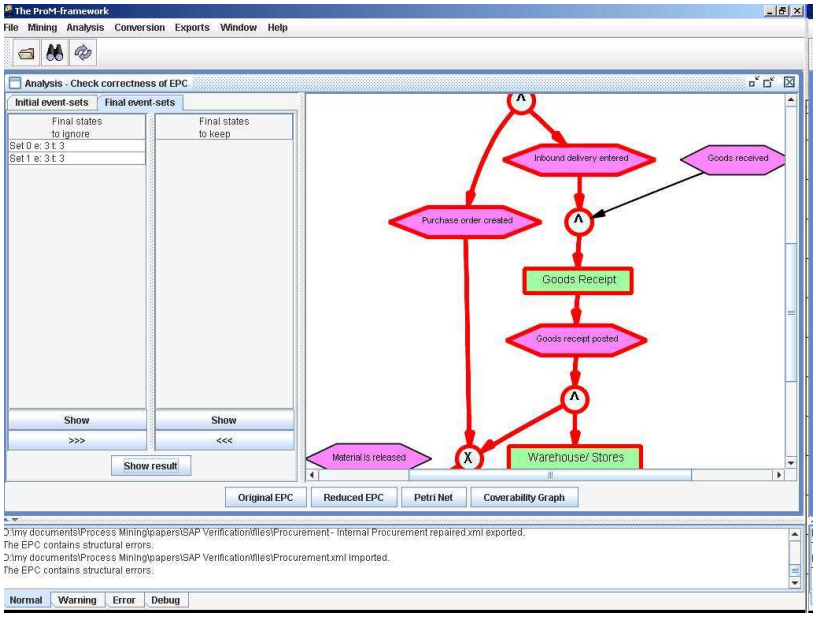

Figure 4: Erroneous "Internal Procurement"

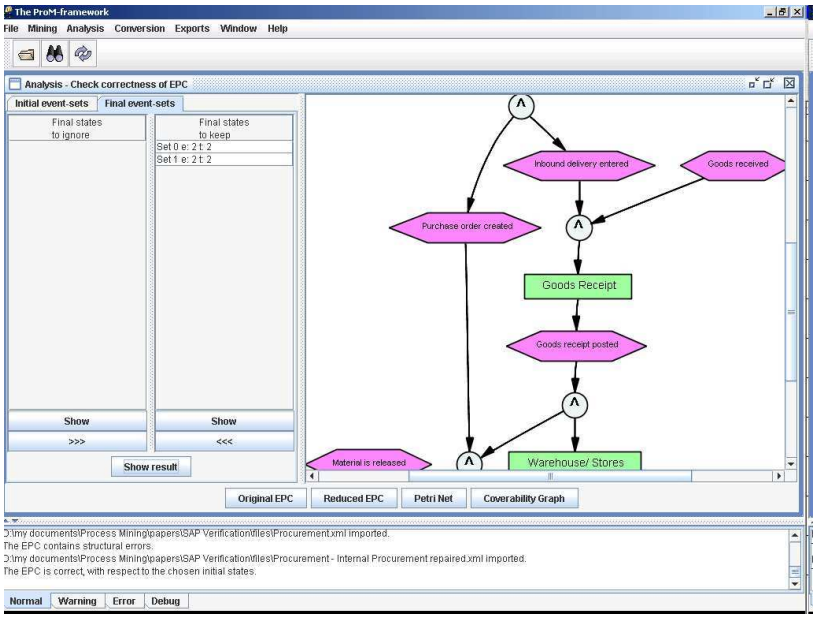

Figure 5: Repaired "Internal Procurement"

ence models, it is a good idea to use one model to create another one. The new model is then customized in such a way that it fits the needs of the new organization better. Figure 8 shows a screenshot of the ARIS toolset, showing two models, namely "Q-notification with Complaint Against Vendor" on top and "Internal Quality Notification" below. These two models are exactly alike, except that in the top-model, a vendors complaint score can be updated. Here, re-use has been applied correctly.

In Figure 9, two models are shown for which the re-use was performed incorrectly. The model on the left hand side represents the handling of a "Service Order" and on the right hand side it represents the handling of a "Maintenance Order". They are very similar, except that the latter does not make a distinction between maintenance at a customer site and at an internal site. Both models however, contain the same mistake, which results from re-using one reference model to create the other model. When services are to be entered, the rightmost event called "Services are to be Entered" occurs. However, when that is the case, due to the XOR-split in front of it, the function "Overall Completion Confirmation" will never be able to execute. Solving this problem requires a good understanding of the modelled situation since many correct solutions are possible. It is important to realize that the changes made to the original model do not introduce the error. The error appears in both models.

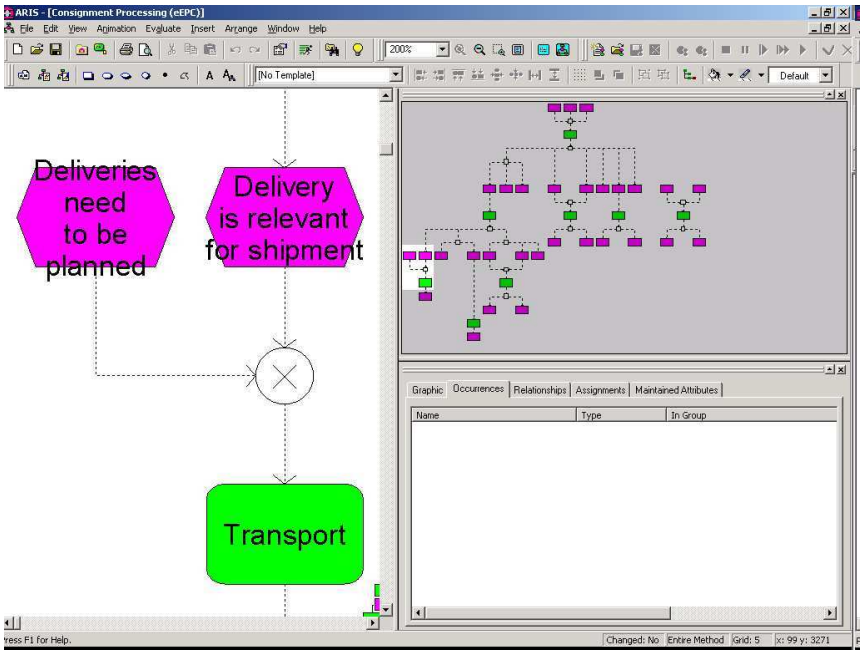

Figure 6: Events joined as XOR ($\times$)

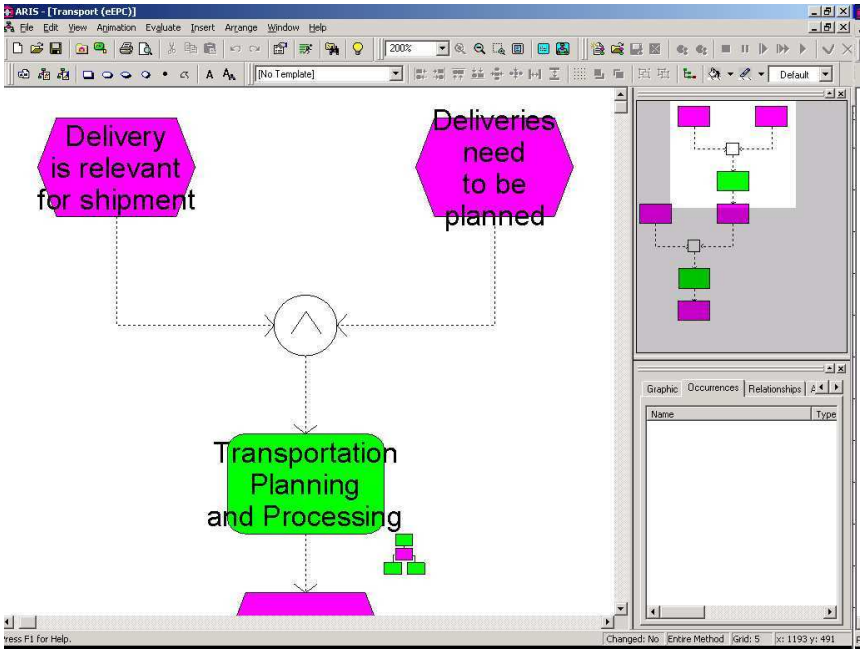

Figure 7: Events joined as AND ($\wedge$)

Table 1: Table of results for the procurement module

Module name	Result	Implication of the problem
Internal Procurement	I	Payments can be done for goods never received.
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Warehouse stores	C	
Pipeline Processing	C	
↪ Invoice Verification	C	
↪ Pipeline Withdrawal	C	
Materials and External Services	S	An invoice can be paid for ordered goods (not services) that have not yet been delivered.
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Service Entry Sheet	C	
↪ Transportation	C	
↪ Warehouse/Stores	C	
Procurement on a Consignment basis	C	
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Warehouse/Stores	C	
Procurement via Subcontracting	I	An invoice that is received twice will be paid twice.
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Provision of Components	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Transportation	C	
↪ Warehouse/Stores	S	When materials are simultaneously placed into the stock and removed from it, erroneous behavior occurs. Operational procedures should avoid this.
Return Deliveries	C	
↪ Invoice Verification	C	
↪ Outbound Shipments	C	
↪ Quality Notification	C	
↪ Shipping	C	
↪ Warehouse	C	
Source Administration	C	
↪ Outline Purchase Agreements	C	Redundant objects are present.
↪ RFQ/Quotation	C	

6 Conclusion and future work

Although we only looked at a small subset of the entire reference model database, we can draw some important conclusions. First of all, it seems that problems are more easily introduced into larger models than into smaller ones. The reason that we did not find many problems in low level models can probably be explained by the fact that these models are typically very small and thus easy to understand. Although this may seem trivial, it shows that even reference models should be kept small and understandable to the designers that use them.

When the smaller models that are usually correct, are connected by higher level models, errors are easily introduced as well. As we saw in Section 5, these errors can lead to severe complications, such as invoices being paid twice. Second, when the same, or similar events

are used in several modules, special care has to be taken. As we saw for the events with respect to shipments, there was no consensus about the use of them in different modules.

Finally, the errors we found with our verification approach were all trivial to repair. Therefore, we feel that the use of such a verification tool in the early stages of process modelling, or reference model development would greatly improve the effectiveness and applicability of these models in later stages.

At this moment, we see two interesting questions that we will follow up on. The first is the question of the involvement of the designer in the verification process. In Section 4, we have shown that the designer is involved in the verification process. However, some decisions were made by the verification tool itself (for example which reduction rules to use). It would be interesting to know to what extent designers want to be involved in the verification process, maybe up to the point where they can specify their own operational semantics for the models under consideration.

Secondly, the verification method selected relies on behavioral properties of the model under consideration. We are interested in a verification method that would include structural properties as well. Although in our analysis performance was never an issue, it could be for larger, more complicated models.

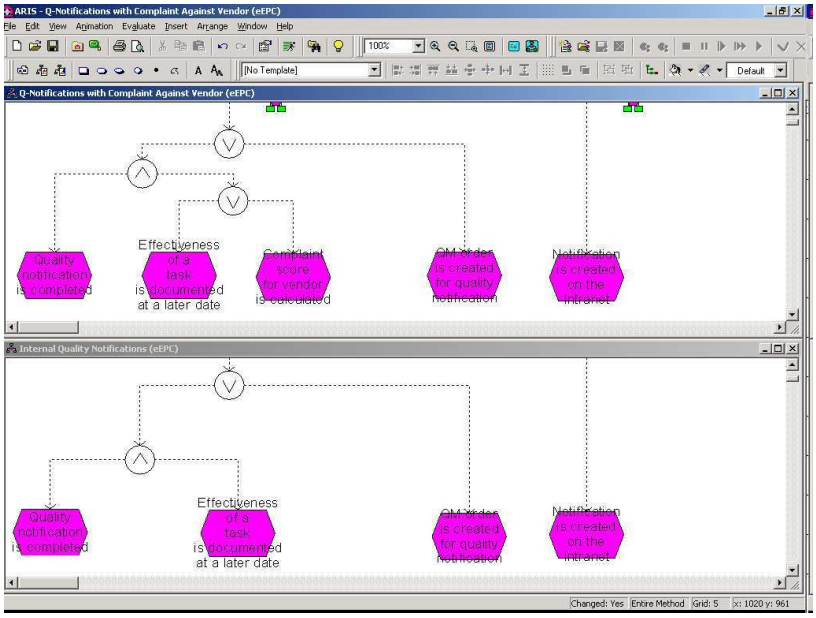

Figure 8: Correct re-use of reference models

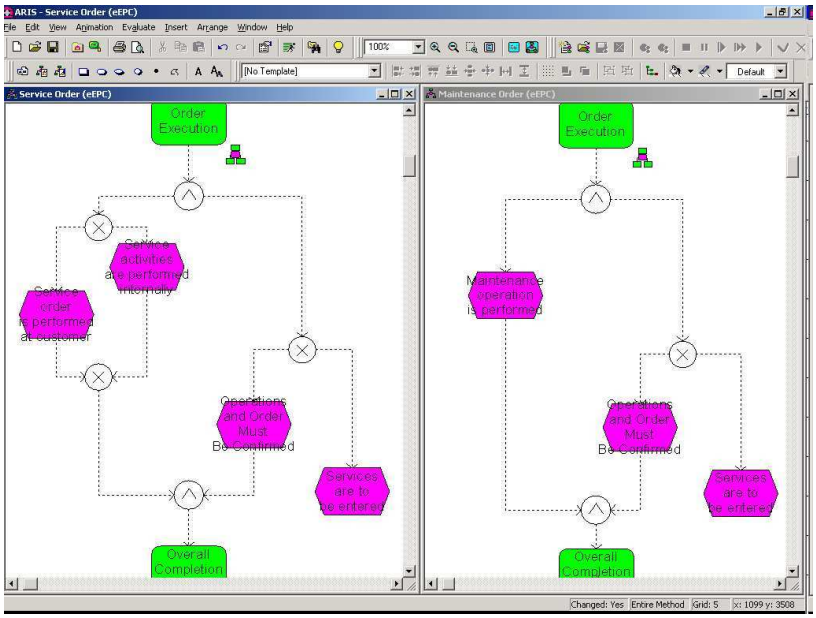

Figure 9: Erroneous re-use of reference models

References

[Aal99] W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.

[Aal00] W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, Berlin, 2000.

[ADK02] W.M.P. van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Nüttgens and F.J. Rump, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, November 2002. Gesellschaft für Informatik, Bonn.

[AH00] W.M.P. van der Aalst and A.H.M. ter Hofstede. Verification of Workflow Task Structures: A Petri-net-based Approach. *Information Systems*, 25(1):43–69, 2000.

[AH02] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.

[AHV02] W.M.P. van der Aalst, A. Hirnschall, and H.M.W. Verbeek. An Alternative Way to Analyze Workflow Graphs. In A. Banks-Pidduck, J. Mylopoulos, C.C. Woo, and M.T. Ozsu, editors, *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02)*, volume 2348 of *Lecture Notes in Computer Science*, pages 535–552. Springer-Verlag, Berlin, 2002.

[Ber] P. Bernus. *GERAM: Generalised Enterprise Reference Architecture and Methodology*.

[CK97] T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.

[DA04] J. Dehnert and W.M.P. van der Aalst. Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, 13(3):289–332, 2004.

[DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

[DJV05] B.F. van Dongen and M.H. Jansen-Vullers. Verification of SAP reference models. In *Business Process Management 2005*, volume 3649 of *Lecture Notes in Computer Science*, pages 464–469. Springer-Verlag, Berlin, 2005.

[DMV+] B.F. van Dongen, A.K.A. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A new era in process mining tool support. In *Application and Theory of Petri Nets 2005*.

[DR01] J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.R. Dittrich, A. Geppert, and M.C. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068 of *Lecture Notes in Computer Science*, pages 157–170. Springer-Verlag, Berlin, 2001.

[DVA05] B.F. van Dongen, H.M.W. Verbeek, and W.M.P. van der Aalst. Verification of EPCs: Using reduction rules and Petri nets. In *Conference on Advanced Information Systems Engineering (CAiSE 2005)*, volume 3520 of *Lecture Notes in Computer Science*, pages 372–386. Springer-Verlag, Berlin, 2005.

[FL03] P. Fettke and P. Loos. Classification of Reference Models - a methodology and its application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.

[Fra99] U. Frank. Conceptual Modelling as the Core of Information Systems Discipline - Perspectives and Epistemological Challanges. In *Proceedings of the America Conference on Information Systems - AMCIS '99*, pages 695–698, Milwaukee, 1999.

[HSV03] K. van Hee, N. Sidorova, and M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In W.M.P. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 335–354. Springer-Verlag, Berlin, 2003.

[Kin04] 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.

[KNS92] 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.

[KT98] G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.

[LR99] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.

[LSW98] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri Nets 1998*, volume 1420 of *Lecture Notes in Computer Science*, pages 286–305. Springer-Verlag, Berlin, 1998.

[LZLC02] H. Lin, Z. Zhao, H. Li, and Z. Chen. A Novel Graph Reduction Algorithm to Identify Structural Conflicts. In *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-35)*. IEEE Computer Society Press, 2002.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[RA03] M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. QUT Technical report, FIT-TR-2003-05, Queensland University of Technology, Brisbane, 2003.

[Ros03] M. Rosemann. *Application Reference Models and Building Blocks for Management and Control (ERP systems)*, pages 595–616. Springer-Verlag, Berlin, 2003.

[RR98] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

[Sch94] A.W. Scheer. *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.

[Sch00] A.W. Scheer. *Business Process Modelling*. 3rd edition, 2000.

[Sil01a] L. Silverston. *The Data Model Resource Book, Volume 1, A Library of Universal Data Models for all Enterprises*. revised edition, 2001.

[Sil01b] L. Silverston. *The Data Model Resource Book, Volume 2, A Library of Data Models for Specific Industries*. revised edition, 2001.

[SO96] W. Sadiq and M.E. Orlowska. Modeling and Verification of Workflow Graphs. Technical Report No. 386, Department of Computer Science, The University of Queensland, Australia, 1996.

[SO99] W. Sadiq and M.E. Orlowska. Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. In M. Jarke and A. Oberweis, editors, *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, Berlin, 1999.

[SO00] W. Sadiq and M.E. Orlowska. Analyzing Process Models using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.

[VA00] H.M.W. Verbeek and W.M.P. van der Aalst. Woflan 2.0: A Petri-net-based Workflow Diagnosis Tool. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets 2000*, volume 1825 of *Lecture Notes in Computer Science*, pages 475–484. Springer-Verlag, Berlin, 2000.

Transformation of BPEL Processes to EPCs

Jan Mendling[1], Jörg Ziemann[2]
[1]Vienna University of Economics and Business Administration, Austria
jan.mendling@wu-wien.ac.at
[2]Institute for Information Systems, University of Saarland, Germany
ziemann@iwi.uni-sb.de

Abstract: The Business Process Execution Language for Web Services (BPEL) is frequently used to implement business processes on a technical level. Yet, as BPEL is also very much related to business logic, BPEL process definitions have to be communicated to business analysts, whether for approval or for process re-engineering. As BPEL does not offer a graphical notation, an automatic transformation to a graphical language like Event-Driven Process Chains (EPCs) is required. In this paper, we present a transformation of BPEL to EPCs. We first define a conceptual mapping from BPEL to EPCs which provides the foundation for a transformation program from BPEL to EPML. Furthermore, we present the concepts used in our transformation program which are also applicable for transformations from block-oriented BPEL to other graph-based process languages.

1 Introduction

Various languages have been proposed for business process modelling focusing on different aspects including business documentation, formal analysis, service composition or choreography [MNN04]. Recently, Web Service composition is gaining increasing attention as a technology to define business processes building on Web Services. The Business Process Execution Language for Web Services (BPEL4WS or BPEL) [ACD+03] is such a language for the definition of executable processes composed of Web Services. Yet, BPEL does not define a graphical notation for its modelling primitives. Accordingly, it is a problem to communicate BPEL process definitions to business analysts when their approval is needed. Basically, this problem stems from a difference between suitable presentations of business processes to business and technical staff.

There is several research available that advocates a transformation between process modelling languages of these two different levels (see e.g. [zMR04, LGB05]). Such an approach is also applicable in order to communicate BPEL processes as Event-Driven Process Chains (EPC) [KNS92]. EPCs are especially well suited to serve as a target for a mapping from BPEL. Firstly, the graphical notation of EPCs is standardized which facilitates understandability. Secondly, as EPCs are well understood by business analysts, because they are frequently used to represent business requirements, e.g. in the context of SAP with the SAP Reference Model [KT98]. Furthermore, there is extensive tool support

for modelling with EPCs. This allows for a simple reengineering of BPEL processes that have been made available as EPC models. Finally, there is a standardized interchange format for EPCs available called EPC Markup Language (EPML) [MN05] that can serve as the target format of a transformation program. In this paper, we present a transformation of BPEL to EPCs. After an introduction into both languages in Section 2, we define a conceptual mapping from BPEL to EPCs (Section 3). This mapping builds the foundation for a transformation program from BPEL to EPML. We continue with a discussion of implementational issues that arose while writing the transformation program, in particular how block-oriented BPEL control flow can be mapped to a graph-based EPC representation (Section 4). Furthermore, we present related research in Section 5 and conclude the paper with an outlook on future research.

2 BPEL and EPCs - An Introduction

BPEL is an executable language to specify Web Service composition. That means that BPEL builds on a set of elementary Web Services to define a more complex process that is also accessible as a Web Service. BPEL offers several concepts of which we briefly sketch those that are relevant for the proposed mapping to EPCs. More details on activities and handlers will be explained in the context of the mapping. For a comprehensive overview refer to the specification [ACD+03].

- *Variables:* In BPEL variables are used to store workflow data and messages that are exchanged with Web Services. Variables have to be declared in the header part of a BPEL process.
- *PartnerLinks:* Partner links represent a bilateral message exchange between two parties. Via a reference to a `partnerLinkType` the `partnerLink` defines the mutual required `portTypes` of a message exchange: it holds a `myRole` and a `partnerRole` attribute to define who is playing which role. `PartnerLinks` are relevant for *basic activities* that involve Web Service requests.
- *Basic Activities:* Basic activities define the operations which are performed in a process. These include operations involving Web Services like the `invoke`, the `receive`, and the `reply` activity. There are further activities for assigning data values to variables (`assign`) or `wait` to halt the process for a certain time interval. Figure 1 shows a code fragment from the example given in the BPEL spec [ACD+03] which includes `invoke`, `receive`, `reply`, `wait` activities.
- *Structured Activities:* BPEL offers *structured activities* for the definition of control flow, e.g. to specify concurrency of activities (using `flow`), alternative branches (e.g. via `switch`), or sequential execution (`sequence`). These structured activities can be nested. Beyond that, `links` can be used to specify synchronization constraints similar to control flow arcs. In Figure 1 sequence activities are nested in a flow activity to define the control flow.

- *Handlers:* There are different handlers in order to respond to the occurrence of a fault, an event, or if a compensation has been triggered. Handlers are declared in the header part of a BPEL process (not shown in Figure 1).

```
001 <process name="purchaseOrderProcess"
002         targetNamespace="..."
003         xmlns="..."
004         xmlns:lns="...">
...
044    <sequence>
045      <receive partnerLink="purchasing"
046               portType="lns:purchaseOrderPT"
047               operation="sendPurchaseOrder"
048               variable="PO">
049      </receive>
050      <flow>
051        <links>
052          <link name="ship-to-invoice"/>
053          <link name="ship-to-scheduling"/>
054        </links>
055        <sequence>
056          <assign>
057            <copy>
058              <from variable="PO" part="customerInfo"/>
059              <to variable="shippingRequest"
060                  part="customerInfo"/>
061            </copy>
062          </assign>
063          <invoke  partnerLink="shipping"
064                   portType="lns:shippingPT"
065                   operation="requestShipping"
066                   inputVariable="shippingRequest"
067                   outputVariable="shippingInfo">
068            <source linkName="ship-to-invoice"/>
069          </invoke>
070          <receive partnerLink="shipping"
071                   portType="lns:shippingCallbackPT"
072                   operation="sendSchedule"
073                   variable="shippingSchedule">
074            <source linkName="ship-to-scheduling"/>
075          </receive>
076        </sequence>
077        <sequence>
078          <invoke  partnerLink="invoicing"
079                   portType="lns:computePricePT"
080                   operation="initiatePriceCalculation"
081                   inputVariable="PO">
082          </invoke>
083          <invoke  partnerLink="invoicing"
084                   portType="lns:computePricePT"
085                   operation="sendShippingPrice"
086                   inputVariable="shippingInfo">
087            <target linkName="ship-to-invoice"/>
088          </invoke>
089          <receive partnerLink="invoicing"
090                   portType="lns:invoiceCallbackPT"
091                   operation="sendInvoice"
092                   variable="Invoice"/>
093        </sequence>
094        <sequence>
095          <invoke  partnerLink="scheduling"
096                   portType="lns:schedulingPT"
097                   operation="requestProductionScheduling"
098                   inputVariable="PO">
099          </invoke>
100          <invoke  partnerLink="scheduling"
101                   portType="lns:schedulingPT"
102                   operation="sendShippingSchedule"
103                   inputVariable="shippingSchedule">
104            <target linkName="ship-to-scheduling"/>
105          </invoke>
106        </sequence>
107      </flow>
108      <reply partnerLink="purchasing"
109             portType="lns:purchaseOrderPT"
110             operation="sendPurchaseOrder"
111             variable="Invoice"/>
112    </sequence>
113 </process>
```

Figure 1: Code snippet of the example given in the BPEL spec [ACD+03]

We present EPCs as captured by the EPML format [MN05] that also serves as the target of our transformation program. EPML offers the traditional EPC elements (see Figure 2): *functions* for modelling activities, *events* to represent pre- and post-conditions of functions, *connectors* to describe different joins and splits, e.g. for concurrent or alternative branches of a process, *hierarchical functions* and *process interfaces* to specify sub-processes. These elements can be connected with control flow arcs. Furthermore, we will use *participant* elements and *data fields* in our mapping from BPEL. The first one captures organizational or human resources involved in the process, the second describe data elements. Both these elements can be connected to function elements via so-called relations. For more details on EPCs and EPML refer to [MN05].

Figure 3 illustrates the mapping from BPEL to EPML by giving the example purchase order process of the BPEL specification (see Figure 1 and [ACD+03]) as an EPC business

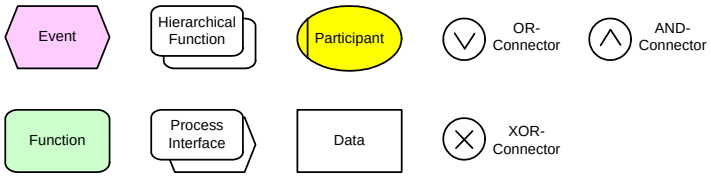

Figure 2: EPC symbols used in the mapping to BPEL

process model. The grey columns highlight the concurrent sequences that are nested within a `flow` activity in BPEL. The process part on the right-hand side captures the fault handler that has been modelled for the process. Yet, it needs to be mentioned that EPCs are not able to express explicit termination semantics - this would be required in order to correctly map fault handlers to EPCs. As you can see in Figure 3, there is an OR join waiting for the fault handler to complete. Standard EPCs do not offer a cancellation concept which could be used to represent termination semantics more appropriately. For more details on this topic refer to [MNN05]. Basic activities map to function-event blocks, that may have relationships with data fields (capturing BPEL variables) or participants (representing partnerRoles of a partnerLink). The example shows that BPEL defines complex business semantics that need to be understood by a process owner or a business analyst. This illustrates the need for an automatic transformation from BPEL to EPCs.

3 Mapping from BPEL to EPCs

As non-control flow elements of EPCs do not have a formal semantics and BPEL still includes some ambiguities (see e.g. [MSW+04]), it is important to explicitly define the purpose of the mapping and to explicate the resulting design principles. Our focus is to provide for a graphical representation of BPEL processes as EPCs in order to communicate the process dynamics to business analysts. This leads to the design principles that are applicable for the proposed mapping:

1. There should be no restriction of the constructs used in the BPEL models. We only assume the BPEL process models to be compliant with the BPEL specification in order to make the mapping work.

2. The EPC visualization focuses on the dynamic behavior of the BPEL model. Elements of a BPEL model that represent static information are represented in the EPC only if they help business analysts to understand the process logic. This implies that partnerLink declarations, variable declarations, and correlation sets as such are not addressed by the mapping. Still variables and partnerRoles of partnerLinks are represented when they are involved with the execution of an activity, e.g. when a receive activity writes an incoming message to a variable.

3. In order to present EPCs in a way that business analysts are familiar with, the in-

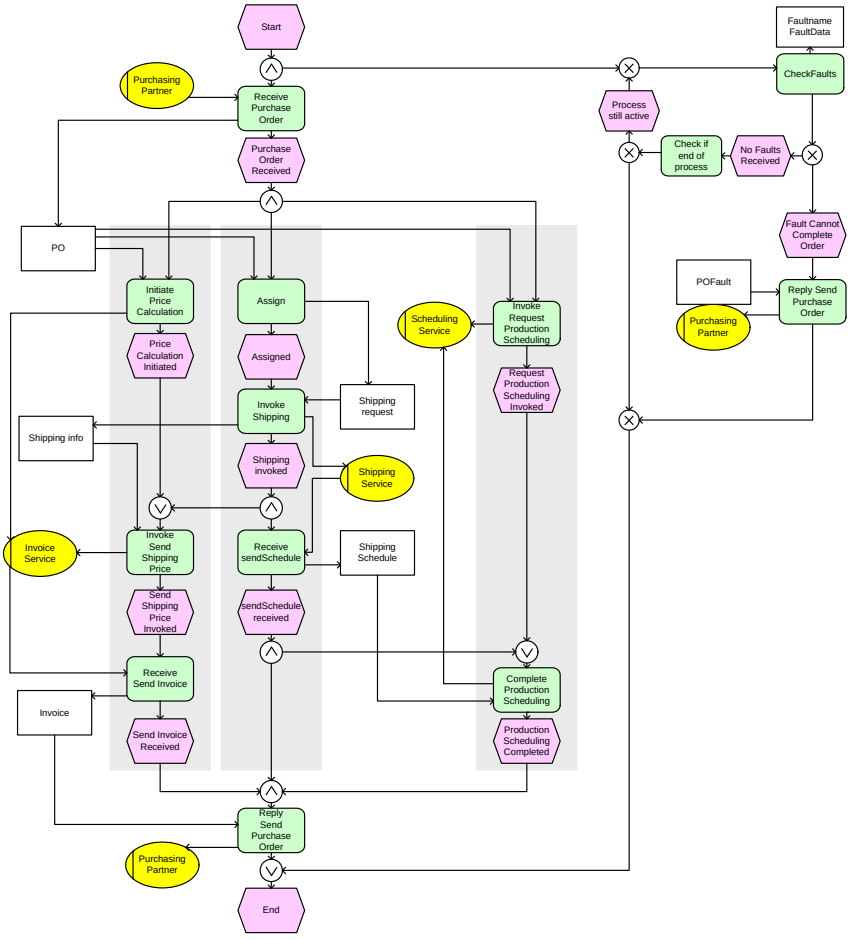

Figure 3: Example Process of BPEL Specification as EPC, compare Figure 1. The grey area highlights the sequences nested in the flow activity [ACD$^+$03].

vention of BPEL specific EPC constructs is not intended here. Yet, BPEL specific parameters may be contained in the resulting EPML file, as long as these attributes do not affect the graphical EPC model. Such parameters can be written to EPML attributes which may be annotated to most elements of an EPML file.

4. BPEL constructs should be transformed to blocks of EPC elements that offer equivalent semantics. EPC elements get names that are generated from the names of the corresponding BPEL elements.

5. It is a point of discussion whether BPEL handlers should be included in generated EPC models. We include the mapping in this paper being aware that business analysts might not be interested in them, and EPC models would be more compact leaving them out. Still, the handler related fragment can easily be deleted from the EPC model.

Mapping of basic activities: There are two aspects that have to be considered for most of the mappings. First, all basic and structured activities may be target or source of links. The BPEL `flow` activity may define multiple links that represent synchronization constraints between a source and a target activity. We will discuss this concept in the context of the `flow`. Second, basic activities map to an EPC function-event block in the general case. Structured activities determine the control flow between these function-event blocks. Beyond such a block, the mapping may generate additional data fields and participants in the EPC representation. We illustrate the mappings in the following.

Invoke, Receive, and Reply: All these three activities are related to Web Service interaction. All of them specify the attributes `partnerLink`, `portType` and `operation`. In the example of Figure 3 the process is instantiated when a purchase order is received from a purchasing partner via a `receive` activity. At run-time the process engine maps `partnerLink` and `portType` to actual endpoints that can be used in the message exchange. All these three activity types map to a function-event block whose names are built from the type of the activity and its `operation` attribute. Accordingly, a receive with `operation` *Purchase Order* yields the EPC function name *Receive Purchase Order*. The three activities involve messages that are read from input and written to output variables. These are mapped to EPC data fields that are connected via a directed relation to the function element. The relation points to the data field if the variable is written, and to the function in the other case. The name of the data field holds the name of the variable involved. Furthermore, a participant element is generated whose name is taken from the `partnerRole` of the `partnerLink`. Figure 4 illustrates the mapping. For the `invoke` activity two cases have to be distinguished. A **synchronous invoke** is similar to the execution of a remote function with in- and out-parameters: the control flow is continued only after the result of the Web Service invocation is received. This implies that synchronous invocations are connected with two data fields representing the input and the output variable. In contrast, the **asynchronous invoke** does not wait for the answer of the remote Web Service, accordingly there is only an input variable to be represented in the EPC model. A optional correlation element inside an `invoke` activity is not transformed, compensation and fault handler can be attached to the activity and will be described in

detail later. The representation of **receive** is similar to `invoke`. Each `receive` is connected to only one data field which receives the incoming message. **reply** on the other hand has a data field for the outgoing message.

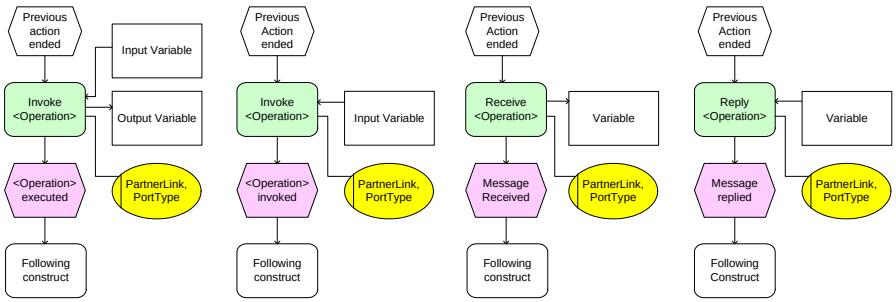

Figure 4: BPEL Web Service related basic activities mapped to EPCs

Other basic activities: Figure 5 shows the EPC representations of the other basic activities. All of them, with `empty` as an exception, are mapped to an function-event block. The **wait** activity is connected to a data field that specifies the time to be waited. Time may be specified either as a duration (e.g. for 1 hour) or by a point in time (e.g. until 2:00pm). The subsequent event occurs when the specified time has arrived. The **terminate** activity immediately terminates all activities of a business process and is followed by an end event of the process.[1] The **assign** activity is used to set the value of variables. The new value can stem either from another variable, from an expressions or from constants. Inside of an `assign` activity one or more `copy` operations specify which values are assigned. Each variable involved is represented in EPCs by a data field. The EPC function is followed by an event that represents the end of the `copy` operation. Another basic BPEL activity, the **empty** activity, does not yield a function-event block. Yet, it has to be included in the mapping if there are `source` and `target` links connected to it (see `flow` activity). The **throw** activity writes a fault to the fault variable of the current scope signalling an exception. This triggers the fault handler (explained in the subsection on handlers). Furthermore, the **compensate** activity is represented by a function-event block. It starts compensation which is also discussed in the context of BPEL handlers.

Mapping of structured activities: BPEL contains five structured activities to define the control flow; those include `sequence`, `switch`, `pick`, `while` and `flow`. A **sequence** contains one or more activities as nested child elements that are executed sequentially in the order they are listed. This sequence is mapped to a sequence of EPC elements corresponding to the elements nested in the BPEL sequence. The **while** activity is used to repeat a basic or structured actitity as long as a specified condition holds true. This is mapped to an XOR join followed by a function to check the condition. An XOR split leads to two events: if the *condition is true* event is triggered, the nested branch is executed another time. Otherwise, navigation continues with the activity subsequent to the `while`. The

[1] This mapping is basically a work-around as end events have implicit termination semantics in EPCs and a cancellation concept is missing in standard EPCs.

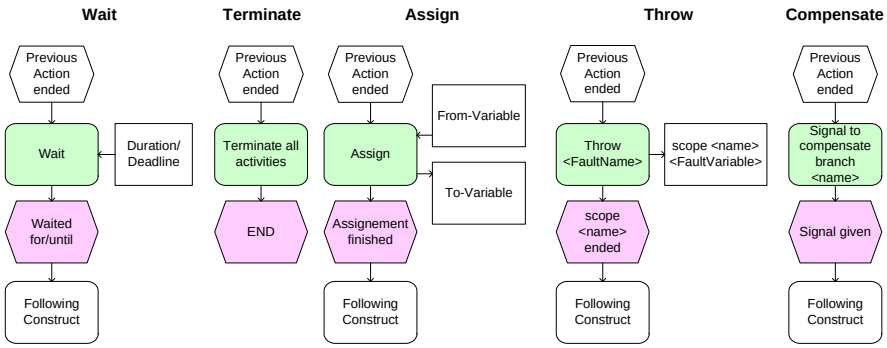

Figure 5: Further BPEL basic activities mapped to EPCs

switch activity consists of one function that evaluates an expression and, depending on the result, one of alternative branches is activated. The EPC representation of `switch` consists of a function for checking the condition followed by a block of alternative branches between an XOR split and an XOR join. The `pick` activity has some similarities to the `switch`. Yet, instead of evaluating an expression it waits for the occurrence of one out of a set of events and executes the associated activities. These events may be related to time or to message receipts. Syntactically, the `pick` maps to the same control flow elements as the `switch`. In the case of `OnMessage` conditions the message is specified with non-control flow elements similar to a `receive` activity. In the case of an `OnAlarm` event the time is modelled similar to the `wait` activity. Each alternative event is followed by nested activities merged with an XOR join. The **flow** construct enables modelling of concurrent activity branches. In EPCs concurrency is modelled by a block of parallel branches started with an AND split and synchronized with an AND join. There may be further synchronization conditions between activities specified by so-called links: each activity nested in a `flow` can be source and target of multiple links. This means that the target activity has to synchronize with the completion of the source activity. In general, each target link maps to an arc that enters an OR join prior to the target activity. Each source link maps to an AND split after the completion event of the source activity. This mapping was applied in the purchase order process of Figure 3. Additionally, the source activities may contain a transition condition (compare left part of Figure 6). If this condition yields true the subsequent AND split activates the following activity of the own branch as well as the link to the target activity. In the other case the own branch is also continues, but without activating the target link. The non-local semantics of EPC's OR join perfectly match to represent the death-path-elimination specified by BPEL for links with transition conditions. For more on this topic refer to [Ki04].

Mapping of handlers: In BPEL so-called **scopes** are used to declare areas of a process that share correlation sets, fault handlers, event handlers, compensation handlers or variables. Handlers will be mapped to EPCs without the need to generate explicit scope constructs. Variable declarations and correlations sets are not transformed. **Handlers** can be associated with whole BPEL processes, scopes, or single invoke activities. In the fol-

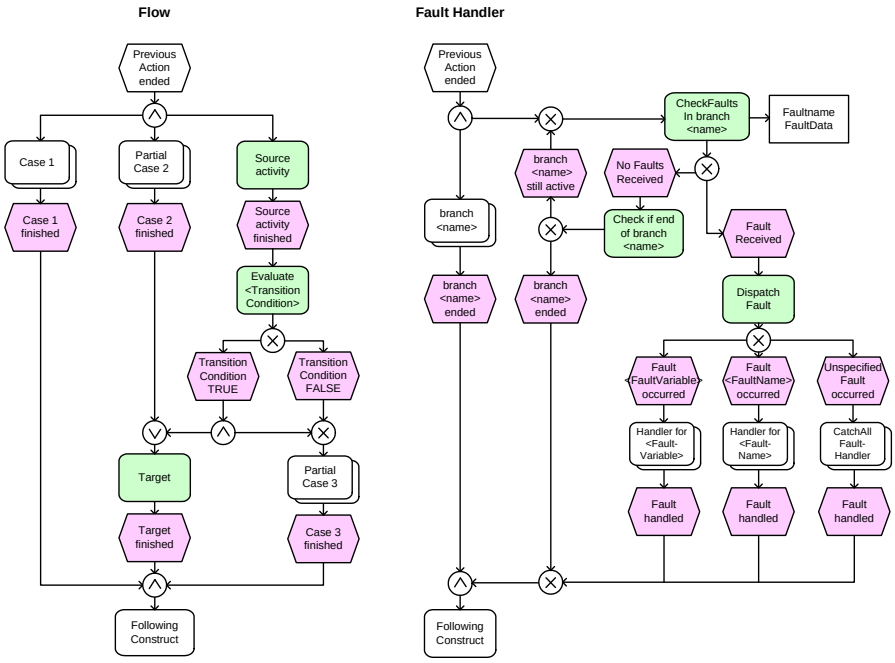

Figure 6: BPEL flow and fault handler mapped to EPCs

lowing we will refer to these three concepts using the term *branch*. Handlers wait for the occurrence of specified events. As a response certain activities are executed which are specified as sub-elements of the corresponding handler. There are two kinds of **event handlers** available. The **onMessage** event handler is mapped to an AND split and AND join that separates the main process from the event handling. The actual handling is mapped to a loop that waits as long for incoming messages as the concurrent main process has not ended. The `OnMessage` handler is associated to non-control flow elements similar to the `receive` activity. Each time a matching message arrives corresponding activities of the handler are executed. If the main process is still active, the loop is re-entered to wait for new matching messages. The `onAlarm` event handler is related to time events. Its mapping is similar to `onMessage`, but the activities to handle the event are executed once at most. Accordingly, there is no loop needed. The time event maps to a function similar to the `wait` activity.

Fault handlers are activated in response to a `throw` basic activity. Throwing a fault stops all processing of the current branch. The fault name and fault data enable the fault handler to identify the fault thrown. Because the borders of scopes are not shown in the EPC, the faults that have to be caught by the attached handler are marked with the name of the branch in which they occur (compare right part of Figure 6). Like the handlers described before, the fault handler waits until either a fault event occurs or the execution of the main branch ends. In the first case a function evaluates which kind of fault occurred and chooses

the corresponding activity. Therefore all BPEL `catch` constructs map to events subsequent to an XOR split. According to the BPEL specification, the event description contains either the name and variable, only the name, or only the variable of a fault. After the fault was handled and not re-thrown to the enclosing scope, the control flow is continued after the branch in which the fault occurred. If a branch does not specify a `catchAll` handler, the so-called implicit fault handler is included. This is mandatory if the scope contains throw activities that have no corresponding handler. The implicit fault handler triggers compensation of all child scopes and re-throws the fault in the parent scope.

Unlike the fault handler a **compensation handler** is only available for invocation if the corresponding activity has completed normally. After this, it can be invoked until the process ends. Therefore, a compensation handler is mapped to a separate EPC process in the EPML file that consists of a loop starting after the activity to be compensated has ended. Inside the loop a function checks if the corresponding compensation signal is thrown, in which case the compensation function is executed. Such signalling can only be represented descriptive in EPCs. Implicit compensation handlers only have to be mapped to EPCs if they contain a child or descendent scope or invoke activity that has a compensation handler.

4 The BPEL2EPML Transformation Program

Building on this conceptual mapping we have started implementing a transformation program called BPEL2EPML in the object-oriented scripting language XOTcl [NZ00], which is an extension of Tcl, using the tDOM package. So far, BPEL processes made up of flow and sequence activities as well as Web Service basic activities can be transformed automatically to EPML. The transformation follows a Flattening strategy[2] as reported in [MLZ05]. For the implementation the following three issues had to be solved: transformation of basic activities, transformation of structured activities, and compliance with EPC syntax rules.

The program processes the structure of nested BPEL activities hierarchically, starting from the top process element. In order to derive a graph-based EPC model, unique IDs have to be generated for each EPC element plus corresponding arcs and relations that reference the correct IDs. The BPEL2EPML program defines a transformation class that holds the $nextId$ of type integer as a class variable. For each activity type, there is a specialized method to generate EPML output. Each time a new EPC element is added to the EPML output, its ID is set to the current $nextId$ which is incremented afterwards. Using this mechanism allows to map each **basic activity** to a function-event block with accompanying data fields and participant elements without a clash of unique ID elements. Yet, there is another mechanism needed to ensure that the function-event block is correctly connected to other function-event blocks via arcs.

The transformation methods for the **structured activities** provide for correct connections between the function-event blocks. Each structured activity method generates the corresponding control structure in which blocks for its child activities can be placed. For each

[2] Pseudo code for the control flow transformation can also be found in [MLZ05].

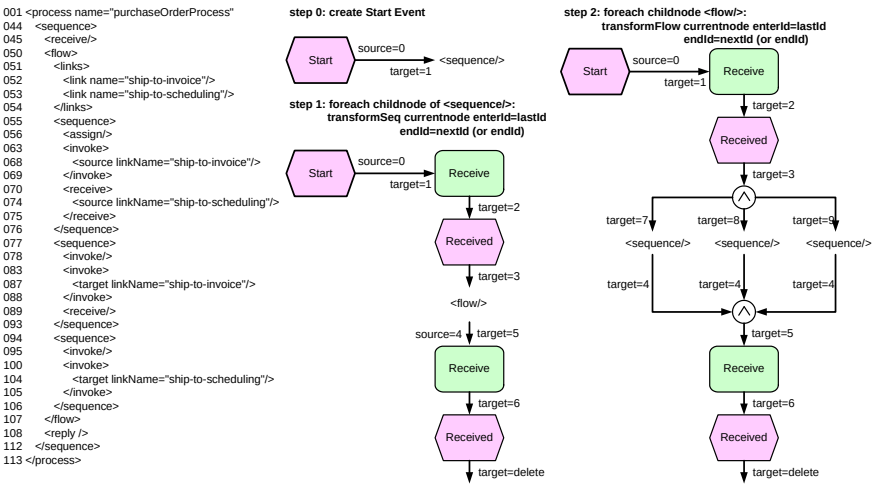

Figure 7: Id generation for BPEL activities

child activity its specialized transformation method is called, whether it is a nested structured or basic activity. There are three parameters that drive the specialized transformation methods: a DOM object holding the BPEL code that still needs to be transformed by the method; an $enterId$ that holds the ID the first EPC element of the nested block must have; and an $exitId$ holding the ID of the subsequent EPC element the nested block must generate an arc to. Consider the BPEL example of the second section which is given in an abbreviated way in the left part of Figure 7. The transformation is started by generating an EPC start event and an arc from it; the $nextId$ is initialized with 1, the $exitId$ is set to $delete$ (step 0). In the next step (step 1) the `sequence` element as the current XML node is transformed by generating EPC fragments for each child of `sequence`. For the last child element the $exitId$ is used as a parameter that the `sequence` received at its invocation. Yet, there is still a problem, because the reply of the example is transformed to a function-event block with an arc from the event to the $exitId$ given. This means an arc is pointing from the *received event* to an $id = delete$. In the next step (step 2) the `flow` is transformed. The IDs of the AND connectors have already been defined as $enterId$ and $exitId$ in the previous step in order to get a coherent EPC graph. This procedure continues until all nested BPEL activities have been processed.

The final arc with target $delete$ illustrates that additional rules have to be encoded to generate EPCs that comply with **EPC syntax rules**. A simple rule is to delete arcs that point to an $id = delete$. Furthermore, a more complex operation is needed to merge e.g. the final events of concurrent branches of a top level `flow` activity to one single end event. Without this operation, the AND join would not have a successor node which is not allowed for an EPC. The last events of the concurrent branches have to be deleted and an end event has to be added after the AND join of the `flow`. This implies that also arcs have to be redirected.

5 Related Research

There are several publications that define transformations between different business process modelling languages, e.g. from UML to BPEL [Ga03], from BPMN to BPEL [Wh04], from EPML to AML [MN04], from BPEL to Petri nets [HSS05] to name but a few. An overview and a comparison of different transformation strategies involving BPEL is reported in [MLZ05]. The merit of our approach is two-fold. First, several of these transformations take a graph-based modelling language as input to generate BPEL. Our contribution is to offer a transformation that facilitates the communication and re-engineering of BPEL process by giving a transformation from BPEL to EPCs as a graph-based language. Furthermore, our contribution is also technical by sketching a transformation for BPEL to graphs that can be also used to generate other graph-based output.

6 Conclusion and Future Work

In this paper, we have introduced a transformation from BPEL to EPCs. Building on a conceptual mapping, we presented a transformation program that is able to generate EPC models as EPML files from BPEL process definitions automatically. Such a transformation helps to communicate BPEL processes to business analysts that are often involved in the approval of business logic. Furthermore, the program can be used for re-engineering of BPEL processes. Finally, the transformation concept is general in such a way that it can be easily adapted to generate output of another graph-based process language that is encoded in XML. In future research we aim to define a profile for EPCs that offers the semantics to be mapped to BPEL. We plan to implement this mapping in a transformation program as well. The transformation from BPEL to EPML defines a starting point for such an endeavor.

References

[ACD+03] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., und Weerawarana, S.: Business Process Execution Language for Web Services, Version 1.1. Specification. BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems. 2003.

[Ga03] Gardner, T.: UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. In: *Proceedings of the First European Workshop on Object Orientation and Web Services at ECOOP 2003*. 2003.

[HSS05] Hinz, S., Schmidt, K., und Stahl, C.: Transforming BPEL to Petri Nets. In: *Proceedings of BPM 2005*. LNCS 3649. S. 220–235. 2005.

[Ki04] Kindler, E.: On the semantics of EPCs: Resolving the vicious circle. In: J. Desel and B. Pernici and M. Weske (Hrsg.), *Business Process Management, 2nd International Conference, BPM 2004*. volume 3080 of *Lecture Notes in Computer Science*. S. 82–97. Springer Verlag. 2004.

[KNS92] Keller, G., Nüttgens, M., und Scheer, A. W.: Semantische Prozessmodellierung auf der Grundlage “Ereignisgesteuerter Prozessketten (EPK)”. Technical Report 89. Institut für Wirtschaftsinformatik Saarbrücken. Saarbrücken, Germany. 1992.

[KT98] Keller, G. und Teufel, T.: *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping*. Addison-Wesley. 1998.

[LGB05] Lippe, S., Greiner, U., und Barros, A.: A Survey on State of the Art to Facilitate Modelling of Cross-Organisational Business Processes. In: *Proceedings of the 2nd GI-Workshop XML4BPM 2005, Karlsruhe, Germany*. 2005.

[MLZ05] Mendling, J., Lassen, K., und Zdun, U.: Transformation strategies between block-oriented and graph-oriented process modelling languages. Technical Report JM-2005-10-10. WU Vienna. October 2005.

[MN04] Mendling, J. und Nüttgens, M.: Transformation of ARIS Markup Language to EPML. In: *Proceedings of the 3rd GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2004)*. S. 27–38. 2004.

[MN05] Mendling, J. und Nüttgens, M.: EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). Technical Report JM-2005-03-10. WU Vienna, Austria. 2005.

[MNN04] Mendling, J., Nüttgens, M., und Neumann, G.: A Comparison of XML Interchange Formats for Business Process Modelling. In: *Proceedings of EMISA 2004 - Information Systems in E-Business and E-Government*. LNI. 2004.

[MNN05] Mendling, J., Neumann, G., und Nüttgens, M.: Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). In: *Proceedings of the 2nd GI-Workshop XML4BPM 2005, Karlsruhe, Germany*. 2005.

[MSW+04] Martens, A., Stahl, C., Weinberg, D., Fahland, D., und Heidinger, T.: Business Process Execution Language for Web services - Semantik, Analyse und Visualisierung. Informatik-Berichte 169. Humboldt-Universität zu Berlin. 2004.

[NZ00] Neumann, G. und Zdun, U.: XOTcl, an Object-Oriented Scripting Language. In: *Proc. of Tcl2k: The 7th USENIX Tcl/Tk Conference, Austin, Texas, USA*. 2000.

[Wh04] White, S. A.: Business Process Modeling Notation. Specification. BPMI. 2004.

[zMR04] zur Muehlen, M. und Rosemann, M.: Multi-Paradigm Process Management. In: *Proc. of the Fifth Workshop on Business Process Modeling, Development, and Support - CAiSE Workshops*. 2004.

Transformation und Mapping von Prozessmodellen in verteilten Umgebungen mit der ereignisgesteuerten Prozesskette

Timo Kahl, Florian Kupsch

Institut für Wirtschaftsinformatik (IWi)
im Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3, Geb. D3 2
D-66123 Saarbrücken
{Kahl|Kupsch}@iwi.uni-sb.de

Abstract: Zwingend notwendig für einen hohen Grad an Interoperabilität zwischen Unternehmen, bei denen Unternehmensmodelle die Grundlage des täglichen Geschäfts darstellen, ist das gegenseitige Verständnis dieser Modelle. Aus diesem Grunde ist zur Optimierung einer Kollaboration ein Modellaustausch zwischen zwei oder mehreren Partnern eine notwendige Voraussetzung. Da in einem kollaborativen Umfeld zumeist mehrere Sprachen und Notationen Anwendung finden, bedingt der Austausch von Modellen zumeist eine Umwandlung von einer Notation oder Sprache in eine andere. Dies geschieht bis dato zumeist manuell. In dieser Ausarbeitung werden deshalb zwei Konzepte diskutiert, das Mapping und die Transformation, die eine teil- und vollautomatisierte Modellumwandlung von einem Format in ein anderes ermöglichen. Beim Mapping geht es um den zuvor beschriebenen Modellaustausch wohingegen die Transformation auf eine Formalisierung der Modelle bis hin zu einer möglichen Ausführung abzielt. Die praktische Umsetzung beider Konzepte wird anhand von zwei Prototypen aufgezeigt.

1. Einleitung

Veränderte Rahmenbedingungen haben Unternehmen vieler Branchen vor eine Vielzahl neuer Herausforderungen gestellt. Diese resultieren vor allem aus der zunehmenden Marktdynamik, dem wachsenden Einfluss der Kunden auf die Gestaltung von Produkten aber auch aus kürzeren Produktlebenszyklen [Sc04]. Eine Reaktion vieler Unternehmen auf diese Entwicklungen ist die Konzentration auf spezifische Kernkompetenzen und eine Auslagerung der nicht zum Kernportfolio gehörenden Unternehmensaktivitäten an Dritte. Dies bedingt jedoch auch eine engere Zusammenarbeit mit anderen Unternehmen und eine Zunahme der Komplexität dieser Zusammenarbeit. Die Gestaltung und das Verhalten kooperativer Organisationsformen wird seit etwa einem halben Jahrzehnt verstärkt mit dem Begriff Collaborative Business (C-Business) bezeichnet [Rö01]. Dabei geht es um eine optimale Gestaltung und Ausführung der unternehmensübergreifenden Zusammenarbeit über die gesamte Wertschöpfungskette hinweg [Ke03]. Eine Voraussetzung für eine

erfolgreiche C-Business-Strrategie und deren Umsetzung ist die Interoperabilität einer Organisationsform. Interoperabilität bezeichnet die Fähigkeit eines Unternehmens mit anderen Unternehmen oder Kunden unter der Prämisse eines optimalen Nutzen-Aufwand-Verhältnis zusammenzuarbeiten [INT05]. Zwingend notwendig für einen hohen Grad an Interoperabilität zwischen Unternehmen, bei denen Unternehmensmodelle Kernstücke des täglichen Geschäfts sind, ist das gegenseitige syntaktische und semantische Verständnis dieser Modelle. Dies zu erreichen erscheint sehr schwierig, da kaum ein Unternehmen eine Modellierungsplattform einsetzt, die verschiedene Modellierungssprachen unterstützt. Folgerichtig ist in den letzten Jahren die Bedeutung der sich über den Lebenszyklus eines Unternehmens erstreckenden Modellierung in vergleichbarem Maße gewachsen wie die Ansprüche an die Interoperabilität dieser Modelle. Die Heterogenität der Modellierungssprachen ist hauptsächlich dadurch bedingt, dass verschiedene Sprachen auf unterschiedliche Modellierungsdimensionen fokussieren und unterschiedlichste Möglichkeiten bieten, die verschiedenen Aspekte eines Unternehmens darzustellen und auszudrücken. Im Zuge dieses Ansatzes wird insbesondere die Prozessdimension fokussiert. Dabei hat sich die Ereignisgesteuerte Prozesskette (EPK) zur semi-formalen Modellierung von Geschäftsprozessen als eine an den Bedürfnissen des Geschäftsprozessmanagements ausgerichtete Adaption eines Petrinetzes etabliert [He02]. Der Grundgedanke des hier beschriebenen Konzeptes ist es, ein Format vorzustellen, das es ermöglicht, Abbildungen von unterschiedlichsten Modellierungssprachen in eben dieses gemeinsame Format zu überführen und somit die EPK interoperabel zu gestalten. Neben diesem Austausch von Prozessmodellen soll darauf aufbauend eine Methode vorgestellt werden, die es erlaubt, die EPK über ein Intermediatorformat in die Workflowsprache XPDL zu überführen. Mit Hilfe dieses Konzeptes wird die vollautomatische Abwicklung kollaborativer Geschäftsprozesse (wie bspw. die Bearbeitung eines Bestellauftrages) möglich, was wiederum eine Effizienzsteigerung der Kollaboration und eine Verbesserung der Interoperabilität zur Folge hat.

2. Transformation und Mapping als zentrale Konzepte zur Gestaltung und Ausführung von Unternehmensmodellen in verteilten Umgebungen

In der wissenschaftlichen Diskussion und praktischen Umsetzung gibt es eine Vielzahl von Ansätzen zur Umwandlung von Unternehmensmodellen (eine Klassifikation findet sich bspw. in [CzHe03]). Im Zuge dieser Ausarbeitung sollen zwei Arten von Umwandlungen näher betrachtet werden. Zum einen wird ein Mechanismus vorgestellt, der es erlaubt, semi-formale Modelle zu formalisieren, um so deren Ausführung zu ermöglichen. Hier wurde in der wissenschaftlichen Literatur eine Vielzahl von Ansätzen diskutiert. Diese lassen sich unterteilen in Konzepte, die eine direkte Transformation einer semi-formalen Beschreibungssprache in eine formale postulieren (bspw. [Zi05]) und solche Ansätze, die eine Transformation über ein Intermediatorformat realisieren. Beide Konzepte finden auch im Zuge der Model Driven Architecture Anwendung, bei der zur Softwareentwicklung zwischen den Ebenen Computation Independent Models (CIM), Platform Independent Models (PIM) und Platform specific Model (PSM) unterschieden wird [MDA]. Im Zuge dieses Konzepts werden die MDA-Ebenen uminterpretiert und an die Bedürfnisse der Unternehmensmodellierung angepasst. In Anlehnung an das Forschungsprojekt ATHENA werden die folgenden drei Ebenen unterschieden [At05a]:

- *Business-Ebene*: Diese Ebene fokussiert auf die Modellgestaltung unter Planungs- und Steuerungsgesichtspunkten. Die Modelle dienen hauptsächlich zu Dokumentationszwecken und sind in der Regel wenig formalisiert. Auf dieser Ebene ist es wichtig, dass der Modellierer nicht durch Formalisierungsvorschriften eingeschränkt wird und eine grafische Repräsentation der Modelle möglich ist. Als Modellierungssprache wird im Zuge des hier vorgestellten Konzepts die erweiterte ereignisgesteuerte Prozesskette (eEPK) verwendet.

- *Technische Ebene*: Die technische Ebene dient als Intermediator zwischen der Business- und Ausführungsebene. In der Regel ist eine direkte Ausführung der Modelle der Business-Ebene nur mit einem verhältnismäßig hohen Aufwand möglich. Aus diesem Grunde werden auf der Technischen Ebene die Sprache der Business-Ebene formalisiert und für die Ausführung notwendige Informationen ergänzt. Als Intermediatorsprache wird hier eine formalisierte Version der eEPK verwendet, die in Kapitel 4 näher vorgestellt wird.

- *Ausführungsebene*: Auf der Ausführungsebene sind die Modelle soweit zu formalisieren, dass sie von Workflow-Engines interpretierbar sind. Im Zuge des hier vorgestellten Ansatzes ist das anvisierte Ausführungsformat die Workflow-Sprache XPDL.

Eine Transformation der Modelle von der Business-Ebene zur Technischen Ebene ist in der Regel nur teilautomatisiert möglich, wohingegen die Überführung von Modellen der Technischen Ebene auf die Ausführungsebene vollautomatisch abläuft [At05a]. In dieser Ausarbeitung ist eine Transformation von Modellen per Definition ebenenübergreifend, muss aber nicht unbedingt sprachübergreifend sein. So kann eine Sprache, die auf der Business-Ebene nicht formalisiert ist mit Hilfe von Modellierungskonventionen in ein formalisiertes Format der Technischen Ebene überführt werden.

Neben der Transformation von Modellen wird insbesondere in der Praxis angestrebt, sprachlich diversifizierte Modelle mit identischem Formalisierungsgrad ineinander zu überführen. Dies wird aufgrund der Tatsache notwendig, dass es insbesondere bei semi-formalen Modellen eine Vielzahl von toolunterstützten Modellierungsmethoden gibt, die auf unterschiedliche Modellierungsdimensionen fokussieren. Beispielhaft werden in dieser Ausarbeitung die Dimensionen Prozess, Produkt, Organisation, Entscheidung und Infrastruktur näher betrachtet. Diese Umwandlung, die innerhalb einer Formalisierungsebene stattfindet, wird im Folgenden als Mapping bezeichnet. Die Differenzierung zwischen Transformation und Mapping ist deshalb notwendig, da im Zuge des Mapping angestrebt wird möglichst alle Dimensionen und Modellinformationen in ein anderes Format zu übertragen. Der Grundgedanke beim Mapping ist es deshalb, ein möglichst generisches Format zur Verfügung zu stellen, das es, kombiniert mit einem methodischen Leitfaden, ermöglicht, Abbildungen von unterschiedlichsten Modellierungssprachen in eben dieses gemeinsame Format zu überführen. Demzufolge zielt das Mappingformat in seiner Funktion als ein Modellaustausch- oder Abbildungsinstrument darauf, die Interoperabilität zwischen kooperierenden Unternehmen, die zugleich unterschiedliche Modellierungssprachen einsetzen, zu verbessern. Bei der Transformation geht es hingegen nicht um einen umfassenden Austausch von Modellinformation, sondern darum semi-formale Modelle in ein formales und ausführbares Modell zu überführen. Dazu ist oftmals eine Anreicherung der Modelle mit

Ausführungsinformationen notwendig. Abbildung 1 visualisiert die zuvor beschriebenen Zusammenhänge.

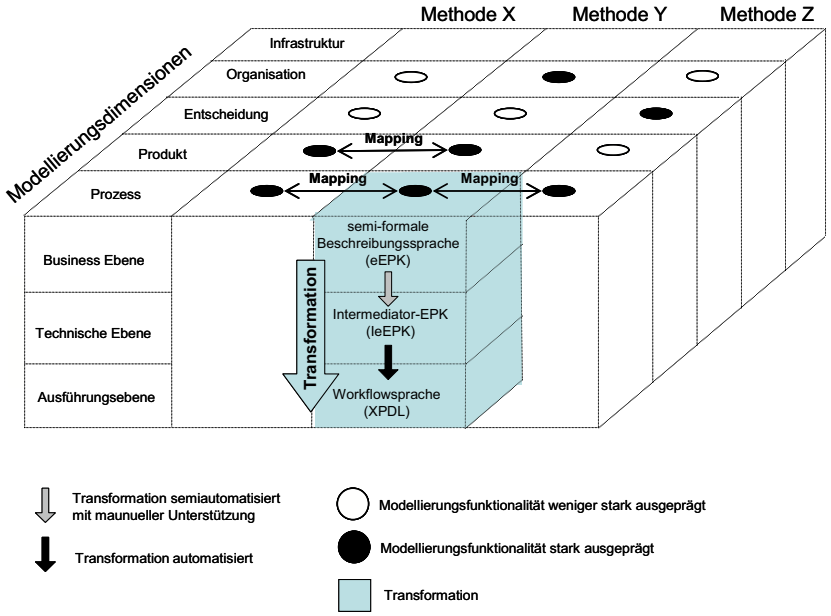

Abbildung 1: Transformation und Mapping

Ziel des hier vorgestellten Ansatzes ist es, basierend auf den Ergebnissen des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten nationalen Forschungsprojektes „P2E2 – Peer-to-Peer Enterprise Environment“ sowie dem europäischen Forschungsvorhaben „ATHENA – Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications“, ein Konzept zu entwickeln, das es ermöglicht die verschiedenen Dimensionen semi-formaler Unternehmensmodelle unterschiedlicher Formate ineinander zu überführen und die Prozessdimension mit Hilfe eines Transformationskonzeptes zur Ausführung zu bringen. Wie Abbildung 1 zeigt, wird nur die Prozessdimension zur Ausführung gebracht, da sie die dynamische Abbildung der anderen Dimensionen ist.

3. POP* als Methode zum Mapping von Unternehmensmodellen

Wie zuvor beschrieben bedingt die Forderung nach einer verbesserten Interoperabilität oftmals den Austausch oder die gemeinsame Nutzung von Modellen. Das Mappingkonzept hat deshalb zum Ziel, ein Instrument zum Austausch von Modellen anzubieten, welches ein generisches Format benutzt, in dem die elementaren Modellierungskonstrukte enthalten sind. Durch ein Mapping von individuellen Modellierungssprachen in dieses Format wird den Untenehmen der Austausch von Modellen eben dieser Modellierungssprache ermöglicht. Das erwähnte Instrument zum Austausch von Modellen ist die POP*-Methode, (nachstehend POP*, was für „Process, Organisation, Product and others“ steht) die im Zuge

des Forschungsprojektes ATHENA entwickelt wird [At05b]. Das hier dargestellte Austauschformat befindet sich aktuell in der Entwicklung und kann sich zukünftig noch ändern.

Das POP*-Metamodell ist, in „Paketen" organisiert, die fünf Dimensionen abbilden und steht unter dem Einfluss verschiedener existierender Ansätze wie BPDM und UEML 1.0. Es existiert je Dimension ein Paket, zuzüglich eines Paketes, das die allgemeinen Konstrukte enthält, die nicht zu einer spezifischen Dimension gehören. Die Dimensionen lassen sich folgendermaßen beschreiben:

1. Das Paket *Allgemeine Konzepte* beinhaltet Konzepte und Beziehungen, die in jeder Dimension angewandt werden können.

2. Die *Organisationsdimension* fokussiert Organisationsstrukturen, menschliche Wesen und deren Interaktion.

3. Die *Prozessdimension* wiederum enthält Konstrukte mit Bezug auf die Aktivitäten, Aufgaben und Prozesse im oder zwischen Unternehmen.

4. Die *Produktdimension* findet Anwendung, um Produktarchitekturen oder Produktstrukturen zu modellieren.

5. *Entscheidungsstrukturen* bezüglich Entscheidungszentren und Entscheidungsaktivitäten werden mithilfe der Entscheidungsdimension dargestellt.

6. Mit den Konstrukten der *Infrastrukturdimension* wird die Modellierung von Infrastrukturen und den von ihnen erbrachten Leistungen ermöglicht.

POP* zielt in seiner Funktion als ein Modellaustausch- oder Abbildungsinstrument darauf, die Interoperabilität zwischen kooperierenden Unternehmen, die zugleich unterschiedliche Modellierungssprachen einsetzen, zu erleichtern. Im aktuellen Stadium ist POP* so ausgelegt, dass es die Interoperabilität der von den ATHENA-Partnern bereitgestellten Modellierungssprachen ermöglicht. Da von ATHENA ein beträchtlicher Teil des Einsatzgebietes von Modellierungssprachen abgedeckt wird, bleibt zu hoffen, dass der Aufwand, weitere Sprachen zu unterstützen nicht all zu hoch ausfällt. Aktuell wird das POP*-Metamodell von einem UML-Klassendiagramm repräsentiert, das in Abbildung 2 abgebildet ist und im Wesentlichen die generische Struktur des POP*-Metamodells beinhaltet. Dies bedeutet im Einzelnen:

- Alle Konzepte im POP*-Metamodell sind Spezialisierungen der Klasse `Object`.

- Alle Beziehungen im POP*-Metamodell sind Spezialisierungen von `Relationship`.

- Alle Eigenschaften eines Objekts oder einer Beziehung sind Spezialisierungen von `Property`.

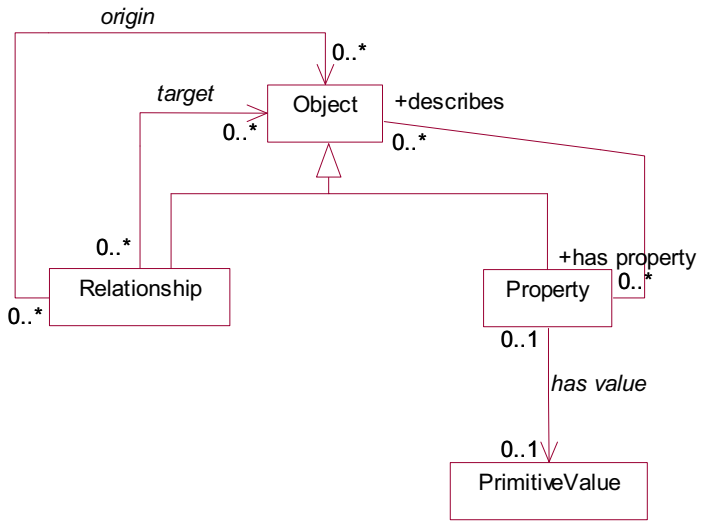

Abbildung 2: Das POP* Kernmodell

Obwohl der Rest des POP*-Metamodells zur Repräsentation auf denselben Formalismus wie das Kernmodell fußt, werden die restlichen Elemente des Metamodells nicht direkt als Spezialisierungen der Kernkonzepte aus Abbildung 2 veranschaulicht. Zu Zwecken der Klarheit und Lesbarkeit werden die folgenden Konventionen in der Notation angewendet:

- Unterklassen von `Objects` (Konzepte des POP* Metamodells) werden als *Klassen* dargestellt;

- Unterklassen von `Relationship` (Beziehungen des POP* Metamodells) werden als *Verbindungen* dargestellt, mit Ausnahme von Beziehungen, die Beziehungen zu anderen Objekten haben (dies bezieht sich augenblicklich nur auf das Konstrukt `Flow`). Im beschreibenden Teil sind alle Beziehungen als "einfache" Klassen mit `origin` (Ausgangspunkt) und `target` (Ziel) als Eigenschaften aufgeführt;

- Unterklassen von `Property` werden in den Darstellungen nicht abgebildet. `Property`-Unterklassen werden im Beschreibungsteil als Eigenschaften der Konzepte die sie beschreiben aufgeführt;

- Wie in den vorangegangenen Aufzählungspunkten dargestellt, `findet ein spezifischer typographischer Stil` Anwendung, wann immer Konstrukte des POP*-Metamodells im Text oder in Bezeichnungen Verwendung finden. Jedoch wird dieses Schriftbild nicht angewendet, wenn die Rede von *Objekten der „realen Welt"* die Rede ist, die von den Konstrukten bezogen werden. Beispielhaft sei hier das Beispiel des Konstrukts `Process` angeführt, das einen *Prozess* im Unternehmen referenziert.

Das POP*-Metamodell ist, wie in Abbildung 3 zu sehen, in „Paketen" organisiert. Es existiert je Dimension ein Paket, zuzüglich eines Paketes, das die allgemeinen Konstrukte enthält, die nicht zu einer spezifischen Dimension gehören.

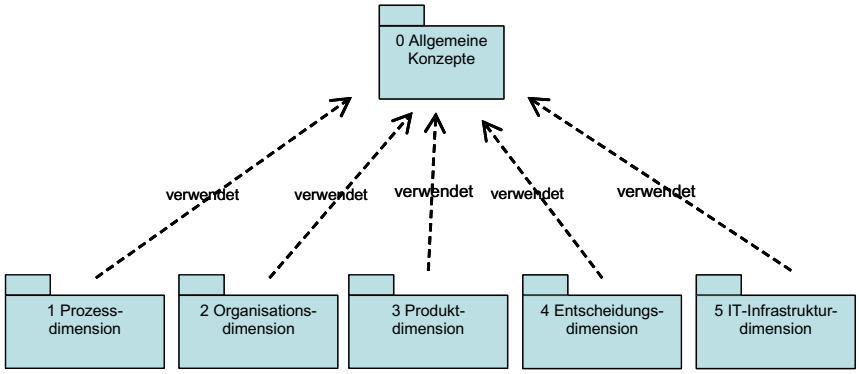

Abbildung 3: Überblick POP* Metamodell

Bislang lag der Schwerpunkt von POP* auf der Prozessdimension, aus diesem Grunde wird in diesem Kapitel exemplarisch für alle anderen Dimensionen das Metamodell der Prozessdimension erläutert.

Die Prozessdimension beinhaltet Konstrukte, die sich auf Aktivitäten, Aufgaben und Prozesse im oder zwischen Unternehmen beziehen. Teilhabe von Objekten des Unternehmens an Prozessen unterschiedlichen Typs wird durch Rollen ausgedrückt. Die Prozesslogik drückt sich in Flüssen und Entscheidungsknoten aus. Eine Übersicht der Modellierungskonstrukte der Prozessdimension ist in Abbildung 4 dargestellt.

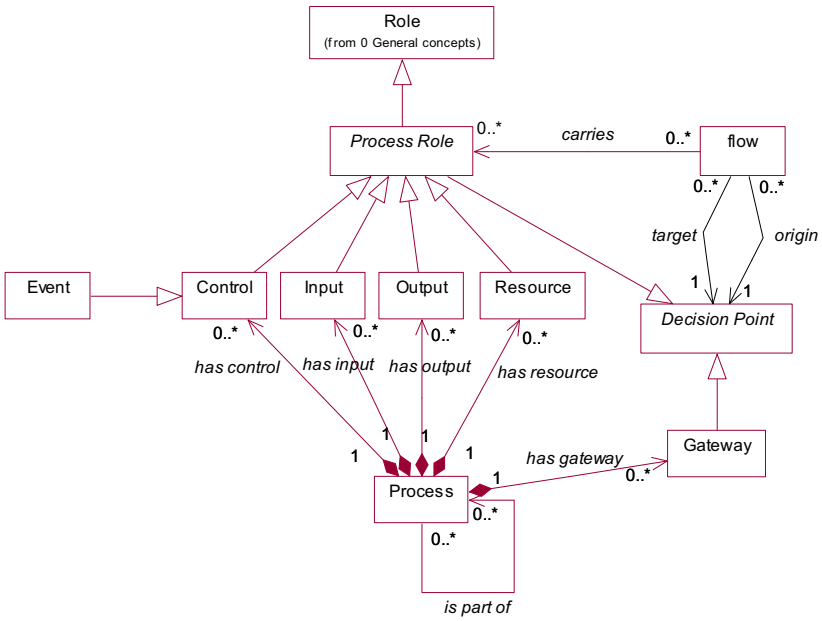

Abbildung 4: Das POP*-Metamodell: Prozessdimension

`Process` wird dabei verwendet um Prozesse, Aufgaben und Aktivitäten jeglicher Art darzustellen. Ein Prozess kann eine Prozessstruktur beinhalten und Teil einer solchen sein (z.B. Projektplan) und zwar durch eine `is part of`-Beziehung. Darüber hinaus können Prozesse zu Sequenzen mit Hilfe von Flüssen und Entscheidungsknoten kombiniert werden. `Decision Point` ist ein generisches Konzept, das – wenn es durch Flows verbunden wird – die allgemeine Prozesssequenz bestimmt. Sowohl `Decision Point` als auch `Process Role` sind abstrakte Konzepte, so dass in einem realen Prozessmodell jeder Ausgangspunkt und jedes Ziel eines Flusses entweder ein `Event`, ein `Control`-Element, ein `Input`-Element, ein `Output`-Element, ein `Resource`-Element oder ein `Gateway`-Element sein muss. Das Verhalten der Entscheidungsknoten wird im Wesentlichen durch die Eigenschaften `in-flow logic` und `conditional continuation` bestimmt, welche wiederum beschreiben, wie multiple Flüsse zu oder von einem Entscheidungsknoten zu behandeln sind. `In-flow logic` spezifiziert wie die eingehenden Flüsse zu kombinieren sind und kann folgende drei Werte annehmen:

1. ‘and’: Alle eingehenden Flüsse müssen aktiviert sein, damit der Entscheidungsknoten aktiviert wird.

2. ‘xor’: Sobald genau einer der eingehenden Flüsse aktiviert wird, wird der Entscheidungsknoten aktiviert

3. ‘or’: Wenigstens einer der eingehenden Flüsse muss aktiviert werden, damit der Entscheidungsknoten aktiviert wird.

Gateway ist einer von zwei Untertypen von Decision Point, der andere trägt die Bezeichnung Process Role. Obwohl beide Typen Bedingungen für die Fortsetzung eines Prozesses ausdrücken, gibt es einen charakteristischen Unterschied. Gateways sind reine Entscheidungsknoten, wohingegen Process Roles zusätzlich zu ihrem Charakter als Entscheidungsknoten auch mit Unternehmensobjekten in Verbindung stehen können via plays role-Beziehungen.

Process Role ist sowohl ein Untertyp von Role als auch von Decision Point, eine Tatsache, die die dualistische Natur des Konzepts veranschaulicht. Wie Roles können alle Process Role Unternehmensobjekte durch die plays role-Beziehung anbinden. Als Decision Point können sie die Bedingungen für Fortsetzung eines Prozesses repräsentieren in welchem die Process Role enthalten ist. Eine Process Role wird spezialisiert in einem von vier Untertypen: Input, Output, Control, oder Resource.

Ein Input gehört zu einem spezifischen Prozess und kann, als Untertyp von Role, repräsentieren, was in den Prozess eingebracht wird. Als ein Untertyp von Decision Point repräsentiert ein Input eine Bedingung, die seinen „Elternprozess" startet. Ein Output gehört ebenfalls zu einem spezifischen Prozess und kann, wiederum als Untertyp von Role, darstellen, was aus einem Prozess hervorgeht (z.B. das Ergebnis). Als ein Untertyp von Decision Point, repräsentiert ein Output eine Bedingung für die weitere Ausführung seines „Elternprozesses".

Auch Resource ist ein Untertyp von Process Role. Eine Ressource gehört zu einem spezifischen Prozess und, in den meisten Fällen, dienen Ressourcen dazu, Rollen und nicht Entscheidungsknoten darzustellen (wobei auch letzteres möglich ist). Als ein Untertyp von Role, werden Ressourcen als Platzhalter für unterschiedlichste Objekte benutzt oder im Prozess erzeugt.

Ein weiteres zentrales Konzept dieses generischen Austauschformats ist das Flow-Konzept. Ein flow ist eine Beziehung zwischen zwei Entscheidungsknoten, Process Roles oder Schnittstellen in einem Prozess. Zusätzlich drückt ein Fluss implizit eine Beziehung zwischen den Prozessen aus, die über Entscheidungsknoten verbunden sind. Im Standardfall ist ein Fluss als Kontrollfluss zu interpretieren. Ein solcher Kontrollfluss wird auf zwei Arten verwendet:

- um zeitliche Abfolgen zwischen Prozessen darzustellen
- um die Auslösung des betroffenen Prozesses kenntlich zu machen

Mit Hilfe dieser generischen Beschreibungselemente ist es möglich mit Pop* eine Vielzahl von Konstrukten aus verschiedenen Sprachen abzudecken. Nur so können die sehr heterogenen Beschreibungsmechanismen unterschiedlicher Sprachen abgedeckt werden. Eines der Hauptprobleme dieser sehr offenen und generischen Beschreibung bleibt der Interpretationsspielraum, der zu Missinterpretationen der unterschiedlichen Konstrukte führen kann. Nachdem in diesem Abschnitt ein generisches Austauschformat zum Mapping von Unternehmensmodellen im Allgemeinen und Prozessmodellen im Speziellen vorgestellt wurde, soll im nächsten Kapitel auf die Transformation von Prozessmodellen eingegangen werden. (das hier vorgestellte Pop*-Austauschformat wurde im Teilprojekt A1 von ATHENA entwickelt und findet sich in der folgenden Spezifikation wider: [At05b])

4. Konzepte zur Transformation von Prozessmodellen mit Hilfe einer formalisierten eEPK

Nach der Erläuterung des Mapping der Prozessdimension, soll hier aufgezeigt werden, wie eine ebenenübergreifende Transformation zu ermöglichen ist. In Anlehnung an das ARIS-Konzept ist die Prozessdimension die zentrale Dimension, die die weitgehend statischen Dimensionen Organisation, Produkt, IT-Infrastruktur und Entscheidung integriert. Somit wird nur die Prozessdimension als dynamische Abbildung der anderen Dimensionen auf die Technische Ebene übertragen und letztendlich zur Ausführung gebracht.

Wie bereits erläutert hat sich zur semi-formalen Modellierung von Geschäftsprozessen die Ereignisgesteuerte Prozesskette (EPK) als eine an den Bedürfnissen des Geschäftsprozessmanagements ausgerichtete Adaption eines Petrinetzes etabliert [He02]. Zentrales Merkmal der EPK bildet die Veranschaulichung der zu einem Prozess gehörenden Funktionen in deren zeitlichlogischer Abfolge. Damit die Regeln und Bedingungen zur Beschreibung der Kontrollflusssteuerung berücksichtigt werden können, kommen Verknüpfungsoperatoren zur Anwendung. Wesentliche Objekttypen der EPK sind damit folgende Elemente:

- *Funktionen.* Eine Funktion transformiert ein Objekt von einem Startzustand in einen Endzustand. Die Transformation dient der Erzielung einer definierten Leistung, die einen wertschöpfenden Bestandteil innerhalb des Geschäftsprozesses darstellt.

- *Ereignisse.* Ein Ereignis ist eine passive Komponente, die Systemzustände bzw. betriebswirtschaftliche Bedingungen repräsentiert, die Einfluss auf den weiteren Verlauf des Geschäftsprozesses ausüben können.

- *Verknüpfungsoperatoren.* Zur Modellierung des Kontrollflusses werden neben gerichteten Kanten konjunktive (AND), disjunktive (XOR) und adjunktive (OR) Konnektoren zugelassen. Diese begünstigen eine im Vergleich zu Petrinetzen erhöhte Anschaulichkeit der Modelle [Sc98].

Da das Transformationsformat die statischen Sichten integriert, ist es notwendig die EPK um Konstrukte anderer Sichten zu erweitern, so dass die erweiterte ereignisgesteuerte Prozesskette (eEPK) die Basis des hier beschriebenen Intermediatorformats ist. Um diese semi-formale Beschreibungssprache in eine Sprache zu überführen, die als Intermediator zwischen formalisierten Workflow-Modellen und semi-formalen Modellen dient und somit eine Tranformation ermöglicht, sind Modellierungskonventionen notwendig. Im Folgenden werden einige allgemeine Konventionen vorgestellt, um im Anschluss daran die Konventionen zur Formalisierung von Modellen zu erläutern.

Die Verwendung von Modellierungskonventionen reduziert die Varietät der Modelle und dient der Vermeidung von Inkonsistenzen sowie einer Verminderung der Anzahl nachträglicher Modellanpassungen [RS02]. Es existieren allgemeine Formalvorschriften für die Ausgestaltung des Kontrollflusses [Ke97] sowie generelle Bestrebungen zur Formalisierung von Syntax und Semantik der EPK [Ru99]. Ebenso wurde die EPK bereits als Spezifikationssprache für Workflows diskutiert [De02] und durch Formulierung von Übersetzungsregeln eine „tragfähige mathematische Basis“ [LSW97] für formale

Geschäftsprozessmodelle geschaffen. Um eine Transformation von EPK-Modellen in ausführbare Workflows zu ermöglichen, sind im Falle einer automatischen Übersetzung besondere Anforderungen an Formalisierungsgrad und Spezifikation des Prozessmodells zu stellen [De02]. Diese Anforderungen sowie Verfahren zu Verifikation von EPK werden seit Jahren in der Forschung thematisiert [Aa99] [Ki04]. In der gängigen Praxis haben sich bei der Modellierung mit EPK neben den ursprünglich von KELLER/NÜTTGENS/SCHEER definierten Konventionen einige weitere Regeln durchgesetzt, die bei einer Modellerstellung zur Anwendung kommen [Ru99]:

- Eine EPK beginnt und endet mit genau einem Ereignis. Das Startereignis triggert den Beginn des Prozesses, das Endereignis definiert einen Zustand, nach dessen Erreichen der Prozess abgeschlossen ist.

- Ereignisse und Funktionen wechseln sich im Ablauf ab. Das Ergebnis der Ausführung einer Funktion wird durch eine Zustandsänderung definiert, welche dem auf die Funktion folgenden Ereignis entspricht.

- Funktionen besitzen genau eine eingehende und eine ausgehende Kante zur Abbildung des Kontrollflusses. Damit ist die Ausführung einer Funktion über eine eindeutige Input-Output Beziehung gekennzeichnet.

- Nach einem Ereignis steht kein OR- oder XOR-Konnektor, da Ereignisse als passive Komponenten keine Entscheidungskompetenz besitzen und damit den weiteren Kontrollfluss der EPK nicht beeinflussen können [KNS92]. Allerdings wird diese Auffassung nicht von allen Autoren vorbehaltlos geteilt [He02].

- Durch Konnektoren verzweigte Pfade werden nur durch gleichartige Konnektoren wieder zusammengeführt. Beispielsweise dürfen die beiden Kontrollflüsse einer XOR-Verzweigung nicht zu einem späteren Zeitpunkt über einen AND-Konnektor vereinigt werden, da diese Bedingung durch die disjunkte Aufspaltung im XOR-Konnektor nicht erfüllt werden kann.

- Werden mehrere Pfade mit einem Konnektor wieder verbunden, darf der Konnektor nur eine auslaufende Kante besitzen, da andernfalls der Kontrollfluss nicht eindeutig beschrieben wäre.

- Direktverbindungen von mehreren Konnektoren (in Form einer Hintereinanderschaltung) sind zulässig, um mehrfache, komplexe Verzweigungen oder Zusammenführungen zu formulieren.

Da semi-formale Modellierungstechniken keine semantische Korrektheit der Prozessmodelle sicherstellen können [EKO96], ist eine Erweiterung der obigen Konventionen erforderlich, um eine formalere Modellerstellung für eine zumindest teilautomatisierte Überführung in ausführbare Workflow-Modelle zu ermöglichen. Da diesem Ansatz die Prämisse zugrunde liegt, die Modelle nicht zu reinen Dokumentationszwecken, sondern als Grundlage zur Ausführung von Prozessinstanzen zu verwenden, zielen die folgenden Konventionen auf die Vereinfachung der Übersetzung in ausführbare Strukturen.

- Innerhalb von Sequenzen darf auf die Formulierung von Ereignissen verzichtet werden. Bei einem linearen Ablauf einer Kette von Funktionen und Ereignissen haben die Ereignisse (sie werden gelegentlich auch als Trivialereignisse bezeichnet) keinen Einfluss auf das Ergebnis der Sequenz. SCHÜTTE unterscheidet in diesem Zusammenhang zwischen Bereitstellungs- und Auslösecharakter von Ereignissen. In einer Sequenz haben Ereignisse damit Auslösecharakter für die folgende Funktion [Sc98]. Sie dürfen daher entfallen, da sie keine zusätzlichen, für die Ausführung relevanten Informationen bereitstellen.

- Bei der Modellierung von Funktionen muss eindeutig zu erkennen sein, ob es sich um manuelle, automatisch-interaktive oder um vollautomatische Funktionen handelt. SINZ konstatiert bei verteilten Anwendungssystemen eine „Spezifikation der Verteilung betrieblicher Anwendungssysteme korrespondierend mit der Zuordnung betrieblicher Aufgaben.“ [FSA96]. Daher sind manuelle und interaktive Funktionen über eine ungerichtete Kante mit genau einer Organisationseinheit zu verbinden, um eine Zuordnung zu einer personellen Ressource herzustellen. Automatische Funktionen müssen über eine ungerichtete Kante mit einem Anwendungssystem oder einem Anwendungssystemtyp verbunden sein, um den unterstützenden maschinellen Aufgabenträger zu identifizieren.

Eine Verwendung des OR-Konnektors (inklusives ODER) ist nicht zulässig. Dies stellt zwar eine wesentliche Restriktion bzgl. des Freiheitsgrades bei der Modellierung dar, die jedoch aufgrund der Eigenschaften der gängigen Workflow-Sprachen für sinnvoll erachtet wird. Beim Setzen eines Konnektors wird implizit festgelegt, ob es sich um eine Zusammenführung oder eine Aufspaltung des Kontrollflusses handelt. Die korrespondierenden Konstrukte der Workflow-Sprachen bestehen in den Operationen SPLIT und JOIN. Analog werden gelegentlich die deutschsprachigen Synonyme „Verzweigung“ und „Synchronisierung“ verwendet [Kr99]. Wie bereits zuvor beschrieben, leistet die EPK in der Modellierung der ARIS Steuerungssicht auf Business-Ebene eine Integration der statischen Sichten [Sc01]. Im Rahmen der Ausführung werden konkrete Workflows gestartet, die aus den Dimensionen Prozess, Organisation und IT-Infrastruktur bestehen. Während eine direkte Übernahme und Anpassung von Prozess- und Organisationsdefinition aus der Organisationssicht möglich ist, ist die IT-Infrastruktur in Form der unterstützenden Anwendungssysteme kein unmittelbarer Bestandteil von ARIS. Zur Beschreibung der IT-Infrastruktur müssen Anwendungssysteme jedoch bereits auf Business-Ebene modelliert in die Prozessdimension integriert werden. Neben Ereignissen und Funktionen werden für die Spezifikation der Aufgabenträger Organisationseinheiten zur Modellierung von Personalressourcen sowie Anwendungssysteme benötigt. Zum Zweck der Aggregation einzelner Prozessteile, beispielsweise um die Detailmodelle nur für interne Organisationseinheiten zugänglich zu halten und nach außen eine höhere Granularitätsstufe zu veröffentlichen, wird der Objekttyp Prozessmodul bereitgestellt. Zur Darstellung des Kontrollflusses finden die Konnektoren „AND“ und „XOR“ sowie gerichtete Kanten Anwendung. In Abbildung 5 ist eine nach obigen Regeln erstellte EPK anhand eines simplifizierten Beispiels der Stornierung einer Bestellung abgebildet. Der Prozess beginnt mit dem Eintreffen einer Auftragsstornierung. Nachdem die Stornierung eingegangen ist, wird die Funktion Stornierung erfassen von einem Mitarbeiter der Organisationseinheit Vertrieb Inland ausgeführt. Der Mitarbeiter benötigt für die Bearbeitung der Funktion das Anwendungssystem KRE-TA.

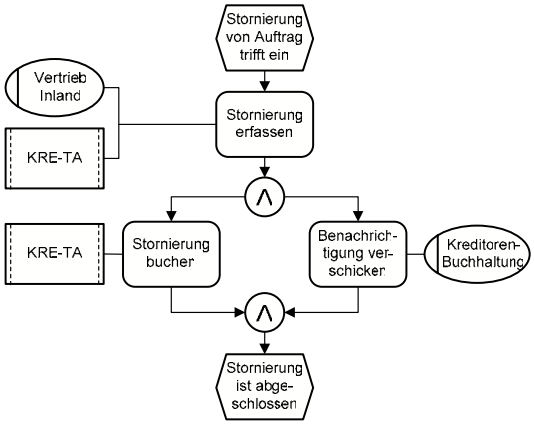

Abbildung 5: Beispiel einer regelkonformen EPK (IeEPK)

Damit handelt es sich um eine automatisch-interaktive Funktion, die durch das Zusammenwirken eines personellen Aufgabenträgers und einer IT-Ressource ausgeführt wird. Nach Erfassung der Stornierung werden die Funktionen Stornierung buchen sowie Benachrichtigung verschicken ausgeführt. Die Buchung geschieht ausschließlich unter Inanspruchnahme des Anwendungssystems »KRE-TA« (vollautomatische Funktion), während das Verschicken der Benachrichtigung ausschließlich von einer Organisationseinheit (und damit manuell) durchgeführt wird. Mit dem Abschluss beider Funktionen ist der Prozess beendet. Mit Hilfe dieser Intermediator-EPK ist es möglich semi-formale Modelle der Business-Ebene über die Technische Ebene auf die Ausführungsebene zu transformieren. Im Folgenden wird diese formalisierte EPK als Intermediator-eEPK (IeEPK) bezeichnet.

5. Prototypische Realisierung der Mapping- und Transformationskonzepte

Nachdem im vorigen Kapitel die konzeptionellen Grundlagen beschrieben wurden, soll hier aufgezeigt werden, wie diese Konzepte im Zuge der Forschungsprojekte P2E2 und ATHENA in Prototypen umgesetzt wurden.

5.1. Prototypische Realisierung des Pop*-Mapping-Konzeptes

Im Zuge des europäischen Forschungsprojektes ATHENA wurde das zuvor beschriebene Pop*-Konzept zum Mapping von Unternehmensmodellen prototypisch realisiert. Um Unternehmensmodelle zwischen Unternehmen zu mappen ist vorgesehen, die Modelle von unterschiedlichen Modellierungsstandards in einen gemeinsamen Standard zu überführen, das POP*-Format. Hierfür wurde eine Transformationsmethode von EPK-Modellen, die mit dem ARIS-Toolset modelliert wurden, in das POP*-Format prototypisch umgesetzt. In

dieser Ausarbeitung sollen die Grundlagen dieses Mapping-Mechanismus kurz beschrieben werden. Sie bilden die Basis einer Java-Applikation, in der diese Konzepte unter der Verwendung des DOM für XML-Transformationen als Basis dienten. Das Programm importiert und exportiert AML (ARIS Markup Language)-Dateien – ein Format, das zur Speicherung von ARIS-Modellen dient.

Hier soll nun lediglich das Mapping grundlegender Elemente der EPK betrachtet werden. Diese Elemente und ihre Beziehungen untereinander werden in der folgenden Abbildung dargestellt.

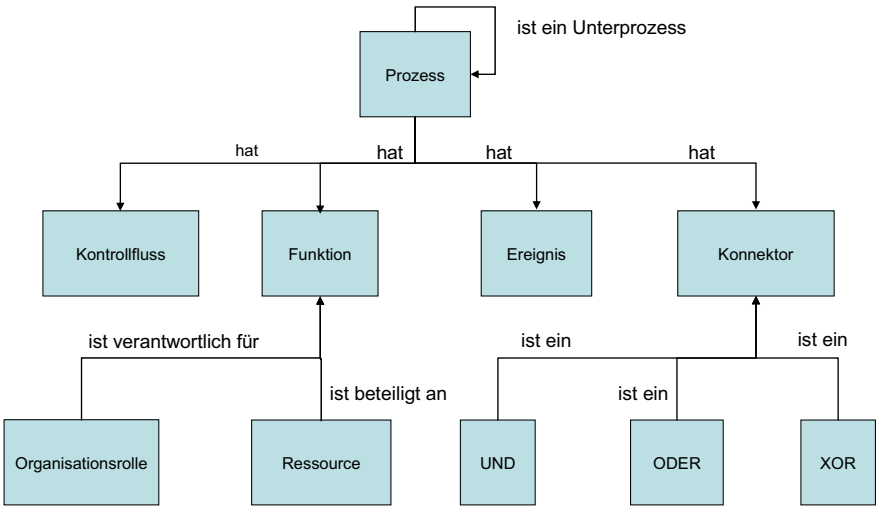

Abbildung 6 Mapping grundlegender eEPK-Elemente

Es können zwei Mapping-Richtungen unterschieden werden. Zum einen sollen mit Hilfe des Pop*-Standards eEPK-Modelle in Modelle anderer Sprachen exportiert werden. Die wesentlich wichtigere Transformationsrichtung für diesen Ansatz ist jedoch der Modellimport. Durch den Import von Modellen, die in anderen Sprachen erstellt wurden, wird es überhaupt erst möglich diese Modelle mit Hilfe der hier vorgestellten Methode auf die Ausführungsebene zu transformieren.

Für den Export ist es notwendig die zuvor beschriebenen Elemente aus der vom ARIS-Toolset generierten AML-Datei zu extrahieren. Die folgende Tabelle bildet die EPK-Quellelemente und die aus der Transformation resultierenden korrespondierenden POP*-Elemente ab.

Tabelle 1: Export von EPK-Modellen

EPK	POP *
Process	Process
Function	Process
AND-Split	Gateway; out-flow logic = AND;
OR-Split	Gateway; out-flow logic = OR;
XOR-Split	Gateway; out-flow logic = XOR;
AND-Join	Gateway; in-flow logic = AND;
OR-Join	Gateway; in-flow logic = OR;
XOR-Join	Gateway; in-flow logic = XOR;
Control Flow (Edge/Link)	Relationship.Flow
Event	State
Organizational Role of a function	ProcessRole/ Role

Um AML-Dateien aus POP*-Daten zu erzeugen, also eine andere Sprache in eine EPK umzuwandeln, werden die folgenden Mappings vorgenommen:

Tabelle 2: Import von EPK-Modellen

POP *	EPK
POP* Process	EPC Process
Decision Point	Connector
Flow	Control Flow
Flow state	Event
Gateway	Connector
Gateway.ConditionalContinuation	Function/Connector
Gateway.inFlowLogic	Connector
Input	Resource attached to function
Output	Only in extended EPC
Output.name	Only in extended EPC
OutputDescription	Only in extended EPC
Process	Process
Process.State	Event
ProcessRole.Name	Organizational Role of a function
Process.hasPart	Subfunction/Hierarchical Function
Resource	Resource
Resource .Activated	-
Resource .ConditionalContinuation	-
Resource .Description	-
Resource .In-flow-logic	-
Resource .Name	Resource (name)
Role	-

Prototypisch umgesetzt wurden die zuvor beschriebenen Implementierungsvorgaben in einem Java-Programm, das es ermöglicht die Modellierungssprache IEM in eine eEPK zu überführen und vice versa. Die Benutzeroberfläche dieses Tools ist in Abbildung 7 zu sehen.

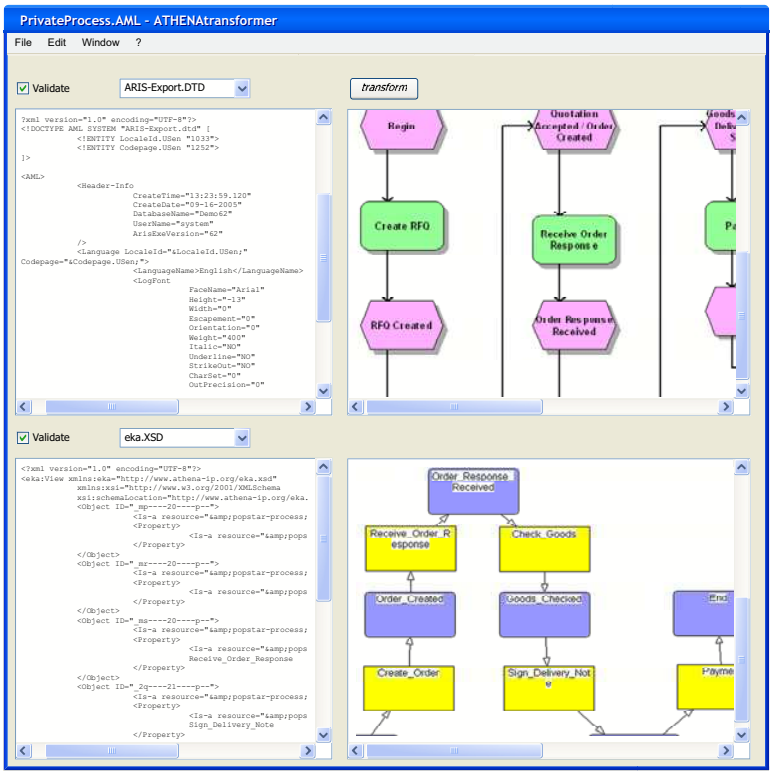

Abbildung 7: Prototyp zum Mapping von Prozessmodellen mit Hilfe von Pop

Die noch nicht vollständig gelöste Aufgabe ist der Abgleich unterschiedlicher POP*- und eEPK-Elemente. Bislang werden nur die grundlegenden prozessbasierten Elemente beider Sprachen transformiert. In den kommenden Entwicklungsphasen müssen weitere Dimensionen (weitere Organisations- und IT-Applikationskonstrukte, etc.) transformiert werden (der hier vorgestellte Prototyp wurde im Projekt ATHENA entwickelt und die Beschreibung und Spezifikation ist in [At05c] zu finden).

5.2. Prototypische Realisierung der Transformation der EPK zu XPDL

Nachdem zuvor ein Prototyp zum Mapping von Unternehmensodellen vorgestellt wurde, soll hier erläutert werden, wie eine ebenübergreifende Transformation prototypisch zu realisieren ist. Der hier beschriebene Ptototyp wurde in dem Forschungsprojekt P2E2 entwickelt. Dabei wäre sowohl die Implementierung eines vollständigen Exportformats wie auch die notwendigen Erweiterungen der Workflow-Engines für das Einlesen noch

lückenhaft spezifizierter XPDL-Dateien mit einem erheblichen Entwicklungsaufwand verbunden gewesen, der den Rahmen des Projektes überschritten hätte. Aus diesem Grund wurde ein alternativer Lösungsweg zur Überführung der Modelle auf der Businessebene in eine Spezifikation gewählt. Dieser sieht vor, die mit dem ARIS Toolset erstellte EPK als Grundgerüst für eine zusätzliche Anreicherung mit den für die WFMS essentiellen Informationen zu verwenden. Über ein Annotationswerkzeug werden die rudimentären Modelle in einem Zwischenschritt um zusätzliche, implementierungsspezifische Attribute erweitert und erst anschließend in die Workflow-Engines eingelesen. Das Annotationswerkzeug fungiert damit als Intermediator zwischen dem betriebswirtschaftlichen und dem informationstechnologischen Prozessverständnis und ist somit als Prototyp der technischen Ebene zuzuordnen. Die Felder der Eingabemasken des Annotationswerkzeugs entsprechen den von der XPDL-Spezifikation geforderten XML-Attributen. Bereits in der Businessspezifikation enthaltene Attribute werden übernommen, können jedoch fallweise vom Benutzer modifiziert werden. Die Implementierung des Tools erfolgte in der Programmiersprache Java. In Abbildung 8 ist die Annotation der Modelle anhand eines Screenshots des Prototypen illustriert.

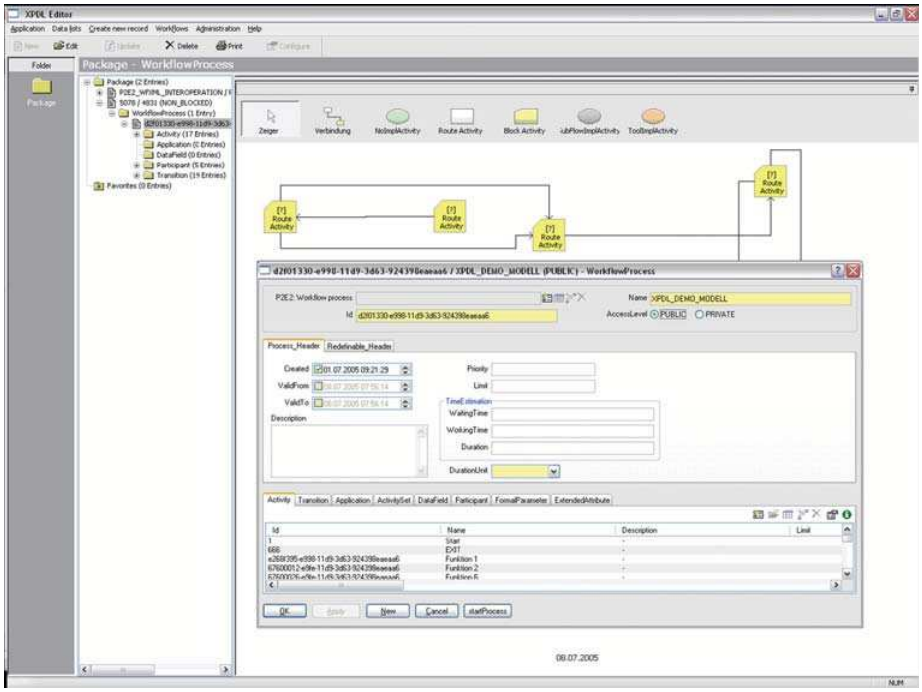

Abbildung 8: Verfeinerung des Workflow-Modells

Das über das Annotationswerkzeug angereicherte Modell kann direkt über die Configuration-API in die Workflow-Engines importiert werden. Zur Konfiguration der Workflow-Engines wurden die Softwareprodukte der Partner um eine Schnittstelle für den XPDL-Import erweitert. Damit können die übersetzten EPK Modelle eingelesen und um interne Konfigurationsparameter erweitert werden. Hier werden die bislang fehlenden, für eine Ausführung jedoch zwingend erforderlichen Detailinformationen wie interne

Bezeichnungen, Datenformate, Anmeldeinformationen etc. sowie eventuell erforderliche topologische Änderungen ergänzt. Zur besseren Nachvollziehbarkeit der vorgenommenen Annotationen besitzen sowohl die Produkte von abaXX als auch die von Carnot eine grafische Modellierungsoberfläche. Am Beispiel der Carnot Process Engine wird in Abbildung 9 gezeigt, wie das durch Unterstützung des ARIS Toolset sowie des Annotationswerkzeugs zum XPDL-Standard konforme Prozessmodell in einer proprietären Notation dargestellt und editiert wird.

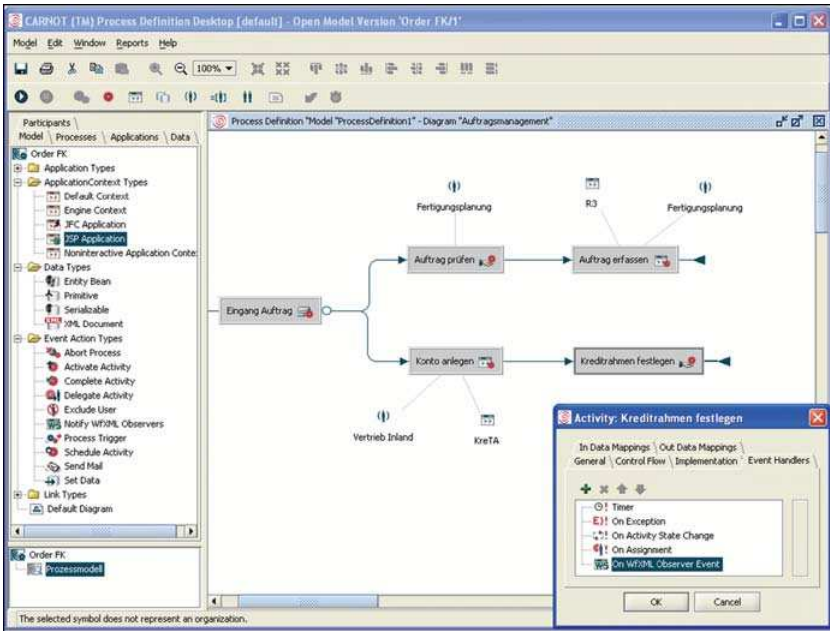

Abbildung 9: Konfiguration der Carnot Process Engine

Der Schwerpunkt des Definitionswerkzeuges der Process Engine liegt jedoch nicht in der grafischen Repräsentation, sondern vor allem in der exakten und vollständigen Konfiguration der Laufzeitumgebung. Einer Aktivität werden zahlreiche Attribute zugeordnet, die deren Ausführungsbedingungen, Parametrisierungen sowie für die Ausführung relevante interne oder externe Ereignisse und Ausnahmen definieren. Das vervollständigte Modell der Konfigurationsdaten wird als sog. Audit Trail in einer beliebigen relationalen Datenbank abgelegt, die über eine JDBC-Schnittstelle verfügt und steht der Process Engine ab diesem Zeitpunkt zur Verfügung.

6. Ausblick

In dieser Ausarbeitung wurden mit dem Mappiug und der Transformation zwei zentrale Integrationskonzepte vorgestellt. Es konnte aufgezeigt werden, wie diese Konzepte konzeptionell zu realisieren und prototypisch umzusetzen sind. Des Weiteren wurde erläutert wie das Zusammenspiel beider Konzepte zur Erhöhung der Intreroperabilität von

Organisationen beitragen kann. Als zukünftige Herausforderung für das Mapping von Unternehmensmodellen ist insbesondere die Ausweitung des Ansatzes auf weitere Sprachen zu sehen. Erst mit der Evaluation des Formats mithilfe mehrerer Sprachen kann dessen Praxistauglichkeit untermauert und evaluiert werden. Im Zuge der Transformation ist insbesondere die Transformation von der Business-Ebene auf die Technische Ebene weiter zu verfeinern und die Möglichkeiten einer vollautomatischen Transformation zu prüfen. Sowohl für das Mapping als auch für die Transformation verspricht die semantische Annotation von Petrinetzen ein erhebliches bislang kaum ausgeschöpftes Potenzial. Selbst unter der Prämisse eines funktionierenden Austauschs von Unternehmensmodellen über mehrere Sprachen, bleibt immer noch das Problem der Missinterpretation von Modell- bzw. Konstruktbezeichnungen. Und auch bei der Transformation würde eine semantische Annotation automatisierbarer Funktionen eine erhebliche Vereinfachung bedeuten.

Literaturverzeichnis

[Aa99] Dehnert, J.: Making EPCs fit for Workflow Management. In: Nüttgens, M.; Rump, F. J. (Hrsg.): EPK 2002 – Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des GI-Workshops und Arbeitskreistreffens (Trier, November 2002). Trier : GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, 2002.

[An03] Andrews T et al.: Business Process Execution Language for Web Services, Version 1.1. May 2003.

[At05a] ATHENA Consortium, Specification of a Cross-Organisational Business Process Model – D.A2.2. ATHENA, Integrated Project - Contract n°:IST-507849, 2005, http://www.athena-ip.org

[At05b] ATHENA Consortium, Report on Methodology description and guidelines definition – D.A1.3.1, ATHENA, Integrated Project - Contract n°:IST-507849, 2005, http://www.athena-ip.org

[At05b] ATHENA Consortium, MPCE Specification – D.A1.5.1, ATHENA, Integrated Project - Contract n°:IST-507849, 2005, http://www.athena-ip.org

[CzHe03] Czarnecki, K. ; Helsen, S.: Classification of Model Transformation Approaches. In: OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, 2003.

[De02] Dehnert, J.: Making EPCs fit for Workflow Management. In: Nüttgens, M.; Rump, F. J. (Hrsg.): EPK 2002 – Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des GI-Workshops und Arbeitskreistreffens (Trier, November 2002). Trier : GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, 2002.

[EKO96] Vgl. Elgass, P.; Krcmar, H.; Oberweis, A.: Von der informalen zur formalen Geschäftsprozeßmodellierung. In: Vossen, G.; Becker, J. (Hrsg.): Geschäftsprozeßmodellierung und Workflow-Management. Bonn [u. a.] : Thomson Publishing, 1996.

[FSA96] Ferstl, O. K.; Sinz, E. J.; Amberg, M.: Stichwörter zum Fachgebiet Wirtschaftsinformatik. Bamberger Beiträge zur Wirtschaftsinformatik (Nr. 36). Bamberg, 1996.

[He02] Heimig, I.: Grammatikbasierte Beschreibung von Geschäftsprozessen. Methodik für das strukturierte Verarbeiten von Modellen. Dissertation. Wiesbaden : Deutscher Universitäts-Verlag, 2002.

[INT05] o. V.: INTEROP: Presentation of the Project. URL http://interop-noe.org/INTEROP/presentation. Abruf am 2005-03-23.

[Ke97] Keller, G.; Teufel, T.: SAP R/3 prozeßorientiert anwenden: Iteratives Prozeß-Prototyping zur Bildung von Wertschöpfungsketten. 2. Auflage. Bonn [u. a.] : Addison-Wesley, 1997, Edition SAP.

[Ke03] Kersten, W.; Kern, E.-M.; Held, T.: Auf dem Weg zur E-Collaboration: Entwicklungslinien im Electronic Business. In: Kersten, W.: E-Collaboration: Prozessoptimierung in der Wertschöpfungskette. Deutscher Universitäts-Verlag, Wiesbaden 2003, S. 8-9.

[Ki04] Kindler, E.: On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In: Desel, J.; Pernici, B.; Weske, M. (Hrsg.): Business Process Management. Berlin [u. a.] : Springer, 2004.

[KNS92] Vgl. Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage »Ereignisgesteuerter Prozeßketten (EPK)«. In: Scheer, A.-W. (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik ; Nr. 89. Saarbrücken : Institut für Wirtschaftsinformatik, 1992

[Kr99] Vgl. Kreplin, K.-D.: Konkordanz englischer und deutscher Begriffe des Workflow Management. WfMC, 1999.

[LSW97] Langner, P.; Schneider, C.; Wehler, J.: Prozeßmodellierung mit ereignisgesteuerten Prozeßketten (EPKs) und Petri-Netzen. In: Wirtschaftsinformatik, 39 1997, Nr. 5).

[MDA] OMG: Model Driven Architecture. Available at http://www.omg.org/mda/

[Rö01] Röhricht, J.; Schlögel, C.: cBusiness: Erfolgreiche Internetstrategien durch collaborative business am Beispiel mySAP.com. Addison-Wesley, München 2001.

[RS02] Rosemann, M.; Schwegmann, A.: Vorbereitung der Prozessmodellierung. In: Becker, J.; Kugeler, M.; Rosemann, M. (Hrsg.): Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung. Berlin [u. a.] : Springer, 2002.

[Ru99] Rump, F. J.: Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten : Formalisierung, Analyse und Ausführung von EPKs. Dissertation. Stuttgart; Leipzig : Teubner, 1999.

[Sc98] Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung. Konstruktion konfigurations- und anpassungsorientierter Modelle. Band 233, Neue betriebswirtschaftliche Forschung. Wiesbaden : Gabler, 1998

[Sc01] Scheer, A.-W.: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen. 4. Auflage. Berlin [u. a.] : Springer, 2001.

[Sc04] Scheer, A.-W., Werth, D., Kahl, T., Martin, G.: Lösungen für das Unternehmen von morgen - Next Generation Business. In: IM - Fachzeitschrift für Information Management & Consulting, imc AG, Saarbrücken, 2004.

[Zi05] Ziemann, J.; Mendling, J.: Transformation of EPCs to BPEL – A pragmatic approach. Akzeptiert für 7th International Conference on the Modern Information Technology in the Innovation Processes of the industrial enterprises, Genua, Italien, September 2005.

Referenzmodellbasiertes Event-Management mit Ereignisgesteuerten Prozessketten

Oliver Thomas, Bettina Kaffai, Peter Loos

Institut für Wirtschaftsinformatik (IWi)
im Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI GmbH)
Stuhlsatzenhausweg 3, Geb. 43.8, 66123 Saarbrücken
[thomas|kaffai|loos]@iwi.uni-sb.de
http://www.iwi.uni-sb.de

Abstract. Events als Kommunikationsinstrument des Marketings gewinnen für Unternehmen zunehmend an Bedeutung. Das Management von Veranstaltungen jeglicher Art stellt ein interdisziplinäres Aufgabenfeld dar, dem sich nicht nur die Praxis, sondern auch Forschungseinrichtungen aus den unterschiedlichsten Bereichen widmen. Da für Events eine sorgfältige Planung im Vorfeld sowie eine möglichst präzise Durchführung von essenzieller Bedeutung sind, kann mit Modellierungssprachen, wie der Ereignisgesteuerten Prozesskette (EPK), ein wesentlicher Beitrag zur systematischen Gestaltung von Event-Management-Systemen geleistet werden. Dieser Artikel spricht demgemäß Empfehlungen zur Anwendungssystem- und Organisationsgestaltung in Form eines EPK-Referenzprozessmodells für das Event-Management aus.

Keywords. Event-Marketing, Event-Management, Geschäftsprozess, Prozessmodell, Prozessmodellierung, Referenzmodell, Referenzmodellierung, Ordnungsrahmen, Ereignisgesteuerte Prozesskette

1 Event im Trend

Events erfreuen sich in Forschung und Praxis seit einigen Jahren vermehrter Aufmerksamkeit. So hat sich ein eigener, speziell auf Events ausgerichteter Dienstleistungszweig entwickelt, an dem sich Event-Agenturen, Messebauer, Künstleragenturen, Ton- und Lichttechniker, etc. an der Organisation und Gestaltung von Veranstaltungen beteiligen. Zahlreiche Untersuchungen bescheinigen dem Kommunikationsinstrument „Event“ großes Potenzial und prognostizieren dem Markt für Events nicht nur quantitatives, sondern auch qualitatives Wachstum [JaSh98; Gold00; GPJC02; Müll03; ZaDr03]. Aufgrund dieses hohen Stellenwerts in der Praxis verwundert es nicht, dass sich auch die Wissenschaft dem Phänomen „Event“ widmet. Nennenswerte Forschungsergebnisse werden vor allem im Marketing und im Tourismus-Management erzielt [Getz97; Nufe02; HeJD02; Lass03; Dren03]. Eine der wesentlichen Erkenntnisse, die aus den Forschungen seit Ende der 1980er-Jahre gewonnen wurde, ist, dass das Management von Veranstaltungen als ein interdisziplinäres Aufgabenfeld zu verstehen ist, das ein effektives und effizientes

Zusammenwirken verschiedenster Partner erfordert. Die strategische Vorbereitung sowie die Planung und Koordination der Durchführung von Events bedürfen einer professionellen Bearbeitung, um ein optimales Zusammenspiel der zahlreichen, an einem Event beteiligten Akteure sicher zu stellen. Eine Unterstützung dieser unter dem Fachbegriff „Event-Management“ subsumierten Prozesse durch moderne Informations- und Kommunikationssysteme ist dementsprechend sinnvoll und bietet zahlreiche Ansatzpunkte [Lupp04, S. 129].

Gleichwohl erfolgen die Gestaltung dieser Event-Management-Prozesse sowie die Entwicklung unterstützender Informationssysteme bislang nicht systematisch, obwohl mit der Informationsmodellierung seit vielen Jahren ein etablierter Ansatz zur Unterstützung eines systematischen Vorgehens der Analyse, Verbesserung, Umsetzung und Steuerung von Geschäftsprozessen besteht [Webe97; Mylo98; Sche99; RoSi01; Kilo02; WaWe02; Hay03]. Es erscheint daher gewinnbringend, Empfehlungen zur Anwendungssystem- und Organisationsgestaltung in Form eines Referenzprozessmodells auszusprechen. Die Konstruktion eines solchen Referenzmodells ist Gegenstand dieses Beitrags.

Die weiteren Ausführungen sind wie folgt gegliedert: Abschnitt 2 gibt zunächst mit der Abgrenzung der Begriffe „Event“, „Event-Marketing“ und „Event-Management“ eine terminologische Grundlage. Die Betrachtung der Event-Management-Prozesse sowie die modellbasierte Entwicklung unterstützender Informationssysteme sind Gegenstand der Abschnitte 3 und 4. Im Anschluss daran werden in Abschnitt 5 die Arbeitsgebiete „Event-Management“ und „Referenzmodellierung“ zusammengeführt, die Anforderungen an ein Referenzmodell für das Event-Management definiert sowie das notwendige Vorgehen für die Erstellung eines solchen Referenzmodells bestimmt (Konstruktionsprozess). Die Konstruktion dieses Referenzprozessmodells erfolgt in Abschnitt 6 (Konstruktionsergebnis). Der Beitrag schließt mit einer kritischen Diskussion der Ergebnisse und einem Ausblick auf zukünftige Forschungsfragen.

2 Vom Event zum Event-Management

Sowohl die alltagssprachliche als auch die wissenschaftliche Verwendung des Begriffs „Event“ sind nicht einheitlich. In den verschiedenen Lebens- und Wissenschaftsbereichen haben sich unterschiedliche Eventbegriffe und -definitionen entwickelt. Dies führt insbesondere in der Wissenschaft bei der Verwendung des Event-Begriffs zu Kommunikations- und Verständnisproblemen. In einer ersten Annäherung können unter Events „temporary occurrences, either planned or unplanned“ [Getz97, S. 4] verstanden werden. Um die Abgrenzung zwischen geplanten und ungeplanten Ereignissen zu verdeutlichen, wird als Betrachtungsgegenstand des Event-Managements meist der Begriff „Event“ um das Attribut „special“ ergänzt. Unter einem „Special Event“ versteht man entsprechend ein „one-time or infrequently occurring event outside the normal program“ [Getz97, S. 4]. Oftmals wird, um eine bessere Handhabbarkeit des Terminus zu gewährleisten, eine Typologisierung von Events vorgenommen. So ist beispielsweise eine eindimensionale Einteilung in „Hallmark Event“ (traditionelles, ortsabhängig stattfindendes Event, wie z. B. Mardis Gras in New Orleans) und „Mega-Event“ (z. B. Olympische Spiele) möglich [Getz97, S. 3–4].

Eine differenzierte, mehrdimensionale Typologisierung von Events kann nach den Dimensionen „Zielgruppe“, „Konzept“ und „Inszenierung des Events“ vorgenommen werden (vgl. Abbildung 1) [Nufe02, S. 39 ff.]. Die erste Dimension fokussiert die in der Literatur häufig anzutreffende Unterscheidung von Events nach deren Zielgruppe. Hier kann zwischen Public Events (unternehmensextern) und Corporate Events (unternehmensintern) unterschieden werden, wobei auch Exhibition Events (Mischform), die beispielsweise Messen und Ausstellungen umfassen, denkbar sind. Nach der zweiten Dimension, der Art der Inszenierung, lassen sich Events in arbeitsorientierte und freizeitorientierte Aktivitäten unterteilen, wobei zwischen beiden Ausprägungen Infotainment-Veranstaltungen eingeordnet sind. Die dritte Dimension bezieht sich auf das dem Event zu Grunde liegende Konzept. Hier wird der Frage nachgegangen, ob der Einsatz des Event-Marketings eher marken- oder anlassorientiert erfolgt, oder ob beide Aspekte zutreffen.

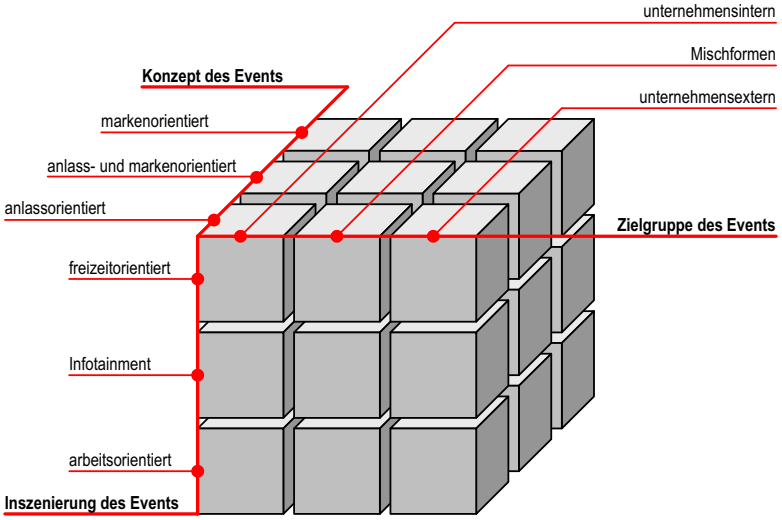

Abbildung 1: Typologisierung von Events [in Anlehnung an Nufe02, S. 40]

Die mit der Planung, Steuerung und Kontrolle von Events verbundenen notwendigen Aktivitäten werden gemeinhin unter dem Begriff „Event-Marketing“ oder „Event-Management“ subsumiert. Bei der Abgrenzung beider Termini wird in der Literatur argumentiert, dass sich Event-Marketing mit den marketingtheoretischen Fundierungen des Phänomens „Event“ befasst und dabei unter anderem Aspekte wie Besuchermotivation und -wahrnehmung oder Imagewirkungen betrachtet. Event-Management hingegen betont eher Fragen der Planung sowie des Qualitäts-, Personal- und Risikomanagements für Events [HeJD02, S. 311].

Bei der Suche nach einer Definition des Begriffs „Event-Management“ zeigt sich, dass in der Literatur keine Einigkeit über den Terminus und den damit in Zusammenhang stehenden Aktivitäten herrscht. Oftmals werden unter Event-Management die organisierenden, überwachenden und steuernden Maßnahmen, die lediglich für die abschließende Durchführung eines Events notwendig sind, zusammengefasst [Erbe02; Hol+03]. Eine

strategische Ausrichtung des Managements mit seinen integrativen Aufgaben wird dabei vernachlässigt. Dies widerspricht einem etablierten Managementbegriff, der den Verantwortlichen außerordentliche Entscheidungsmöglichkeiten einräumt.

Zur Begriffsbildung ist darüber hinaus von Bedeutung, dass Vorhaben zur Planung und Durchführung von Events in der Regel zwei typische Merkmale aufweisen. Sie beginnen *erstens* mit der Definition der Anforderungen, die an das Event zu stellen sind, und enden mit dessen Abschluss. Sie sind somit zeitlich befristet und weisen einen klaren Anfangs- und Endzeitpunkt auf. Bei den Vorhaben handelt es sich *zweitens* vielfach um einmalige Initiativen, an denen unterschiedliche interne und externe Organisationseinheiten beteiligt sind. Aufgrund beider Merkmale weisen Prozesse zur Planung und Durchführung von Events in der Regel Projektcharakter auf. Diese Interpretation von Events als Projekte ist anerkannten Bestimmungen des Projektbegriffs angelehnt, wie sie beispielsweise mit den Beiträgen von FRESE [Fres00], HABERFELLNER [Habe92], MADAUSS [Mada00] oder LITKE [Litk04] vorliegen. Insbesondere MADAUSS [Mada00, S. 499] weist – nach kritischer Prüfung der merkmalsbasierten Projektdefinitionen mehrerer Autoren – die zeitliche Befristung (klarer Anfangs- und Endzeitpunkt) sowie die Einmaligkeit von Vorhaben als eindeutige Projektmerkmale aus. Die Projekteigenschaft von Events wird bei einer Begriffsbestimmung in der Literatur häufig vernachlässigt.

Aufgrund dieser Überlegungen soll daher die folgende Arbeitsdefinition zu Grunde gelegt werden: *Event-Management* umfasst die Koordination aller strategischen, planenden, realisierenden und kontrollierenden Aufgaben und Tätigkeiten zur Durchführung eines Events, basierend auf den Grundsätzen des Event-Marketings und den Methoden des Projektmanagements.

3 Event-Management-Systeme

Um eine möglichst umfassende Unterstützung aller Tätigkeiten und Beteiligten über den gesamten Event-Management-Prozess hinweg zu gewährleisten, sind neben allgemeinen Planungsaktivitäten vor allem Aspekte der Informationstransparenz, des Informationsaustauschs, der Informationsspeicherung sowie der Dokumentations- und Kontrollmöglichkeiten zu betrachten. Proprietäre Softwarelösungen zur Textverarbeitung, Tabellenkalkulation, Projektmanagement oder E-Mail-Kommunikation bieten keinen integrierten Ansatz für das Event-Management. Zwar existieren neben den gängigen Standardanwendungen auch auf spezielle Domänen ausgerichtete Anwendungssysteme, wie z. B. Gastronomie- oder Ticketsysteme, jedoch bieten diese lediglich Spezialfunktionalitäten, wie etwa Kalenderdarstellungen, Lösungen zur Raumplanung, die Möglichkeit der Hinterlegung von Zusatzinformationen oder Lösungen zur Besucherregistrierung [Lupp04]. Eine umfassende informationstechnische Unterstützung des gesamten Event-Management-Prozesses, von der ersten Idee über die Einbindung in die Unternehmensstrategie bis hin zum Abschluss eines Events, ist bislang nicht umgesetzt.

Die Potenziale einer entsprechenden Softwarelösung liegen vor allem in einer größtmöglichen Informations- und Kostentransparenz. Die Effizienz- und Effektivitätssteigerungen, die durch den Einsatz eines entsprechenden Werkzeugs zur Planung, Durchführung

und Kontrolle eines Events entstehen würden, sind analog zum Einsatz entsprechender Systeme im Supply Chain Management zu sehen [ScAT02]. So ist nicht nur eine verbesserte Koordination und Kommunikation der am Prozess Beteiligten, z. B. einer Event-Agentur und einem Event-Dienstleister, zu nennen. Auch der Kunde – als Auftraggeber eines Events – profitiert von einer verbesserten Transparenz. Entscheidungen über mögliche Änderungen lassen sich schneller und kostengünstiger treffen, da Kommunikationswege deutlich verkürzt und so Informationen schneller ausgetauscht werden können.

Darüber hinaus sind Themen wie Controlling oder Risk Management von zunehmendem Interesse für die Planung von Events. Bisherige Ansätze konzentrierten sich auf ökonomische Evaluierungen nach Abschluss eines Events [Clar04, S. 2 f.]. Es wird jedoch oftmals gefordert, kontrollierende Maßnahmen in allen Phasen des Event-Management-Prozesses durchzuführen, um die Nachhaltigkeit eines Events gewährleisten zu können. Daher werden angemessene Möglichkeiten der Dokumentation und die Bereitstellung adäquater Controlling-Methoden benötigt, die nur durch eine entsprechende Werkzeugunterstützung gewährleistet werden können.

Event-Management-Systeme, hier umfassend verstanden als Informationssysteme, die der Unterstützung des Managements von Events dienen, müssen als Vermittler zwischen den betriebswirtschaftlichen Rahmenkonzepten des Event-Marketings und -Managements sowie der Informationstechnik fungieren. Da Event-Management-Systeme damit sowohl eine betriebswirtschaftliche als auch eine technische Ebene berühren, sind sie – wie generell betriebliche Informationssysteme – sehr komplex. Durch Modellkonstruktionen sollte daher versucht werden, handhabbare Artefakte zu schaffen, mit denen die Komplexität dieser Informationssysteme beherrschbar wird.

4 Modellierung von Event-Management-Systemen

Informationsmodelle haben sich als ein Medium zur Überbrückung der Lücke zwischen betriebswirtschaftlichen Problemstellungen und der Realisierung eines Anwendungssystems etabliert [Webe97; Mylo98; Sche99; RoSi01; Kilo02; WaWe02; Hay03]. Die Anwendungsmöglichkeiten von Informationsmodellen reichen vom Softwareentwurf über die Einführung und Konfiguration von Standardsoftware bis hin zum Business Process Reengineering.

Die Konstruktion von Modellen ist aus Gründen ihrer möglichen Wiederverwendung vielfach mit dem Anspruch verbunden, von unternehmensspezifischen Eigenschaften zu abstrahieren. Sie werden daher in unternehmensspezifische Informationsmodelle und Referenzmodelle unterschieden. Der Begriff „unternehmensspezifisch“ charakterisiert hierbei lediglich den individuellen Charakter des entsprechenden Modells, mit dem keine Einschränkung auf rechtlich selbstständige Unternehmen verbunden ist. Aus Gründen einer sprachlichen Eindeutigkeit sollte daher eher von spezifischen Modellen gesprochen werden, um dem Umstand Rechung zu tragen, dass die Spezifität der Modelle nicht ausschließlich aus einem Unternehmenskontext heraus gegeben sein muss, sondern z. B. auch aus einem Projektkontext. Zur Hervorhebung dieses Kontextes kann dann auch von projektspezifischen Modellen gesprochen werden.

Im Gegensatz dazu stellt ein Referenzmodell für die Entwicklung spezifischer Modelle einen Bezugspunkt dar, da es eine Klasse von Anwendungsfällen repräsentiert [Schü98; vBro03; Thom05]. Die Möglichkeit, sich an den fachlichen Inhalten solcher Referenzmodelle orientieren zu können, verspricht den Modellanwendern einerseits die Einsparung von Zeit und Kosten. Andererseits kann durch die Verwendung eines Referenzmodells die Qualität des zu konstruierenden Modells und somit die Qualität der auf Basis dieses Modells entwickelten Software erhöht werden. Der mit diesen Überlegungen verbundene Grundgedanke der Referenzmodellierung, Prozesswissen modellbasiert zu speichern, um es an anderer Stelle wiederverwenden zu können, wird auch aktuell in der Event-Management-Literatur erkannt. So postuliert beispielsweise SCHWANDNER, dass „es fast immer klüger [sei], sich gute Ideen anderen abzuschauen, deren Tipps zu befolgen und dies dann nach eigenen Bedürfnissen zu optimieren“ [Schw04, S. 27]. Gleichwohl liegen im State-of-the-Art keine referenzmodellbasierten Gestaltungsempfehlungen für Event-Management-Systeme vor. Dieser Mangel ist vor allem auf die folgenden Problemfelder zurückzuführen:

- *Mangelnde Prozessorientierung:* Die betriebswirtschaftliche Forschung im Umfeld des Event-Marketings und -Managements vernachlässigt nach wie vor Aspekte des Prozessmanagements für Events: „Less research has focused on special events operational management“ [HeJD02, S. 322]. Sie betont vor allem Fragen der kulturellen, gesellschaftlichen und ökonomischen Auswirkungen von Events. Eine sämtliche Aspekte des Event-Managements integrierende Sichtweise fehlt [Pepe98, S. 612; Tool00, S. 86].
- *Mangel an standardisierten Repräsentationsformen:* Marketingorientierte Arbeiten konzentrieren sich auf die Erklärung von Wirkungszusammenhängen, die in der Regel mithilfe von Marktforschungsstudien erarbeitet werden. Darüber hinaus dominieren in der Literatur beispielhafte Darstellungen und Vorschläge zum Management von Events. Die hierbei verwendeten, wenig standardisierten Repräsentationsformen begrenzen die Aussagekraft der vorgestellten Konzepte und erschweren eine anwendungsfallspezifische Anpassung [Lars03, S. 219–220]. Lediglich vereinzelt trifft man auf Ansätze, welche diesem Sachverhalt entgegenwirken, indem generell anerkannte Methoden, z. B. des Projektmanagements, deduktiv auf den Bereich des Event-Managements übertragen werden [Tool00].
- *Mangel an Modellen:* Vor allem praxisorientierte Untersuchungen, die sich mit der Planung und Organisation von Events beschäftigen, fokussieren einen Zusatznutzen in Form von vorgefertigten Checklisten, Tabellen, Formularen und Roadmaps [Erbe02; Hol+03]. Anschauliche modellhafte Darstellungen, wie sie im Arbeitsgebiet der Informationsmodellierung üblich sind, findet man nur selten.

Die nachfolgenden Untersuchungen versuchen, diese Mängel durch referenzmodellbasierte Gestaltungsempfehlungen für das Event-Management zu beheben.

5 Anforderungen an Referenzmodelle im Event-Management

Das Ziel dieses Abschnitts ist die Definition der wesentlichen Anforderungen an das zu konstruierende Referenzmodell. Zunächst ist eine Analyse des Markts an Referenzmodellen voranzustellen, da das Vorhandensein eines adäquaten, d.h. die definierten Anforderungen erfüllenden, Referenzmodells das Entwicklungsvorhaben obsolet machen kann [BDKK02, S. 42]. Diese Analyse entspricht der Darstellung verwandter Arbeiten.

5.1 Existierende Referenzmodelle in Forschung und Praxis

In der Literatur existieren zahlreiche Referenzmodelle für verschiedene Anwendungsdomänen – zu einer aktuellen tabellarischen Übersicht vgl. FETTKE, LOOS [FeLo03, S. 46f.]. Während frühe Ansätze, wie z.B. das Kölner Integrationsmodell (KIM) [GGGP71], auf die Repräsentation von Aspekten sämtlicher Unternehmen ausgerichtet sind, ordnen die Autoren gegenwärtiger Konstruktionen ihre Referenzmodelle häufig konkreten Wirtschaftszweigen zu. Prominente Beispiele sind das Referenzmodell für industrielle Geschäftsprozesse von SCHEER [Sche97] und das Handelsreferenzmodell von BECKER, SCHÜTTE [BeSc04], die dem wissenschaftlichen Umfeld entstammen.

In der Praxis sind Referenzmodelle einerseits bei Anbietern von Modellierungswerkzeugen und Unternehmensberatungen zu finden. So werden beispielsweise von der IDS SCHEER AG [IDS03b] diverse Referenzmodelle angeboten. Hierbei handelt es sich um Referenzmodelle für den Dienstleistungssektor (Banken, Handelsunternehmen, Kommunalverwaltung, Krankenhäuser, Versandhandel, Versorgungsunternehmen, Versicherungen), die stückorientierte Fertigung (Anlagenbau, Kfz-Zulieferer, Maschinenbau, Konsumgüterindustrie, Möbelindustrie) und die prozessorientierte Fertigung (Chemische Industrie, Papierindustrie). Andererseits existieren umfangreiche Dokumentationen etablierter ERP-Systeme in Form von Referenzmodellen, wie beispielsweise das SAP R/3-Referenzmodell [CuKL98]. Ein Referenzmodell, das im weitesten Sinne dem Themenfeld „Event-Management“ zugerechnet werden kann, ist den Autoren nicht bekannt.

5.2 Notwendigkeit der Konstruktion eines Ordnungsrahmens

Um dem Anspruch einer Wiederverwendung bei der Konstruktion von Modellen zu genügen, müssen Referenzmodelle vielfältige betriebliche Gegebenheiten und deren Interdependenzen beschreiben. Sie werden zudem aus unterschiedlichen Perspektiven betrachtet. Eine überblicksartige grafische Darstellung der Referenzmodelle wird dadurch erschwert. So enthält beispielsweise das Datenmodell des SAP R/3-Referenzmodells mehr als 4000 Entitytypen und das entsprechende Referenzprozessmodell mehr als 1000 Geschäftsprozesse [CuKL98]. In Forschung und Praxis hat sich daher für umfangreiche Referenzmodelle die Verwendung von Ordnungsrahmen bewährt [Sche97; Meis01; BeSc04]. Referenzmodellordnungsrahmen liefern ein navigierbares Verzeichnis, dessen Ordnungsbereiche auf Detailmodelle des Referenzmodells verweisen. Die nachfolgende Erstellung des Event-Management-Referenzmodells wird daher in die Gestaltung des Ordnungsrahmens und die Konstruktion des Referenzmodells unterschieden.

Im Gegensatz zur Erstellung der Detailmodelle werden für die Konstruktion eines Ordnungsrahmens in der Regel keine Modellierungssprachen eingesetzt. Durch die Verwendung frei definierter grafischer Symbole kann der Modellentwickler besonders vielfältige inhaltliche Gesichtspunkte des Referenzmodells verdeutlichen. Sie können zudem dazu beitragen, den Markenzeichencharakter eines Referenzmodellordnungsrahmens zu unterstreichen. Gleichwohl sind in den Sprachportfolios der Modellierungswerkzeuge einiger Anbieter, neben den „etablierten" Sprachen (z. B. ER-Diagramm, EPK) auch „einfache" Modellierungssprachen enthalten, die speziell auf die Konstruktion von Ordnungsrahmen ausgerichtet sind. Im *ARIS-Toolset* wird beispielsweise das Y-Diagramm „für den funktionsorientierten Einstieg in komplexe Referenzmodelle verwendet" [IDS03a, S. 4–7]. Die angesprochene Einfachheit der Sprachen bezieht sich hierbei auf die geringe Anzahl an Sprachelementen und an zwischen diesen Sprachelementen konstruierbaren Beziehungen sowie auf die grafische Repräsentation der Sprachelemente durch elementare geometrische Strukturen, wie Linie, Kreis oder Polygon.

Durch die Zuordnung von Teilen des Referenzmodells zu Verzeichnisbereichen eines Ordnungsrahmens werden die entsprechenden Elemente des Modells nach inhaltlichen Kriterien gruppiert. Das der Konstruktion des Referenzmodellordnungsrahmens zu Grunde liegende Modellobjekt ist das Referenzmodell. Ordnungsrahmen und Referenzmodell stehen daher in einer Makro-Mikro-Beziehung. In diesem Sinne befindet sich ein Ordnungsrahmen immer auf einer „höheren" Aggregationsebene als das durch ihn repräsentierte Referenzmodell. Die Desaggregation von Makromodellen kann auch „innerhalb" des Referenzmodells über mehrere Aggregationsebenen fortgesetzt werden, was sich insbesondere bei umfangreichen Referenzmodellen anbietet. Dies setzt allerdings voraus, dass die Möglichkeit zur Desaggregation in der verwendeten Modellierungssprache als unterstützte Konstruktionstechnik verankert ist. Mit der Ereignisgesteuerten Prozesskette (EPK) wird nachfolgend eine solche Prozessmodellierungssprache zur Repräsentation des Referenzprozessmodells für das Event-Management ausgewählt.

5.3 Modellierungssprache zur Repräsentation der Referenzprozessmodelle

Obwohl die ersten Ideen zur Wiederverwendung von Informationsmodellen mehr als drei Jahrzehnte zurückliegen, wurden bislang kaum „eigene" Modellierungssprachen für die Erstellung und Nutzung von Referenzmodellen konzipiert. Zwei der wenigen Ausnahmen sind die von LANG, TAUMANN, BODENDORF [LaTB96] vorgeschlagenen Referenzprozessbausteine und das Referenzmodellkomponentendiagramm von VOM BROCKE [vBro03, S. 235 ff.]. Die meisten Arbeiten im Arbeitsgebiet der Referenzmodellierung konzentrieren sich auf eine anwendungsfall- oder domänenspezifische Auswahl etablierter Sprachen zur Informationsmodellierung. Das Spektrum der Begründungen zur Auswahl dieser Sprachen reicht dabei von der grundlegenden Orientierung an Paradigmen (z. B. objektorientiert oder nicht-objektorientiert) oder Modellierungsmethoden (z. B. ARIS oder UML) bis hin zum gänzlich unkritischen und unreflektierten Einsatz der Sprachen. Gelegentlich werden Erweiterungen der ausgewählten Modellierungssprachen vorgenommen.

Seit Ende der 1970er-Jahre wurde eine Vielzahl an Modellierungssprachen entwickelt, die der Beschreibung von Prozessmodellen dienen. Zur Konstruktion von Referenzprozessmodellen auf konzeptioneller Ebene hat sich insbesondere im deutschsprachigen Raum die Ereignisgesteuerte Prozesskette (EPK) etabliert [ScTh05]. Referenzprozessmodelle, die unter Verwendung der EPK konstruiert sind, stellen im Forschungsstand der Referenzmodellierung unter anderem NÜTTGENS [Nütt95], KRUSE [Krus96], GEIB [Geib97], LANG [Lang97], REMME [Remm97], SCHEER [Sche97], KELLER, TEUFEL [KeTe99], SCHWEGMANN [Schw99] und BECKER, SCHÜTTE [BeSc04] vor. Die EPK wird nachfolgend zur Konstruktion des Referenzmodells für das Event-Management verwendet.

Die EPK wurde am Institut für Wirtschaftsinformatik (IWi), Saarbrücken, in Zusammenarbeit mit der SAP AG entwickelt [KeNS92]. Sie gilt als zentrale Modellierungssprache der Architektur integrierter Informationssysteme (ARIS) und hat – nicht zuletzt aufgrund ihrer Anwendungsorientierung und einer umfassenden Werkzeugunterstützung – einen hohen Grad der Verbreitung und Akzeptanz in der Praxis gefunden [NüRu02, S. 64]. Sie ist Bestandteil des *ARIS-Toolset* der IDS SCHEER AG sowie des Business Engineering und Customizing des SAP R/3-Systems. In der graphentheoretischen Terminologie ist ein EPK-Modell ein gerichteter und zusammenhängender Graph, dessen Knoten Ereignisse, Funktionen und Verknüpfungsoperatoren sind [ScTh05].

6 Konstruktion des Referenzmodells für das Event-Management

Die nachfolgend erläuterten Modelle wurden im Rahmen des DFG-Projekts „Referenzmodell-gestütztes Customizing unter Berücksichtigung unscharfer Daten“, Kennwort: Fuzzy-Customizing, am Institut für Wirtschaftsinformatik (IWi) im DFKI über den Zeitraum eines halben Jahres entwickelt. Die Modelle wurden mittels Interviews und Workshops in Zusammenarbeit mit drei großen deutschen Event-Agenturen, einem Vertreter der Marketingabteilung eines Automobilkonzerns sowie Mitarbeitern eines international agierenden Messedienstleisters erarbeitet. Neben diesem induktiven Vorgehen zur Erkenntnisgewinnung, bei der eine Vielzahl tatsächlich beobachteter Ablaufstrukturen beschrieben, geordnet und verglichen werden mussten, wurden außerdem – gemäß einer deduktiven Vorgehensweise – Erkenntnisse aus allgemein anerkannten Grundsätzen und Denkmodellen der in der betriebswirtschaftlichen Literatur behandelten „Theorie“ des Event-Managements gewonnen.

6.1 Konstruktion des Referenzmodellordnungsrahmens

Der Ordnungsrahmen für das Event-Management, der in Abbildung 2 dargestellt ist und aufgrund seiner Form nachfolgend auch kurz *Event-E* genannt wird, strukturiert die zur Planung und Durchführung von Events notwendigen Tätigkeiten in einer schlüssigen Abfolge. Der Ordnungsrahmen ist in die fünf Ordnungsbereiche „Event-Strategie“, „Event-Planung“, „Event-Realisierung“, „Event-Kontrolle“ sowie „Projektmanagement“ unterteilt.

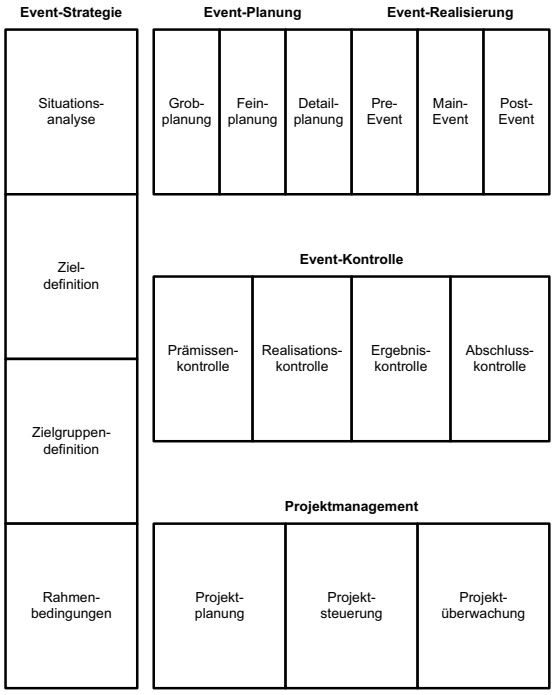

Abbildung 2: Event-E – Referenzmodellordnungsrahmen für das Event-Management

Entsprechend der zeitlich-logischen Abfolge des Event-Management-Prozesses ist zunächst die Event-Strategie zu betrachten. Innerhalb dieser Phase werden alle grundlegenden, das Event betreffenden Fragestellungen in Abstimmung mit der Unternehmens- und Marketingstrategie beantwortet. Die Planungsphase ist die Phase, innerhalb der die zeitliche und räumliche Koordination aller Aktivitäten und Akteure für das Event erarbeitet wird. Die Event-Realisierung umfasst die tatsächliche Durchführung des Events am Veranstaltungsort. Die Event-Kontrolle stellt sämtliche kontrollierenden Methoden und Maßnahmen zu allen Zeitpunkten des Event-Managements bereit. Das Projektmanagement bildet die planerische Wissensbasis für den gesamten Event-Management-Prozess.

Den genannten Ordnungsbereichen sind jeweils Funktionen (auch Phasen genannt) zugeordnet. Diese Funktionen zeichnen sich durch zeitliche und sachlogische Abhängigkeiten aus. Für einige Aktivitäten ist der Abschluss vorhergehender Tätigkeiten oder das Vorhandensein gewisser Dokumente eine notwendige Voraussetzung für deren Durchführung. Die fünf Ordnungsbereiche sind nicht als voneinander unabhängige Prozesse zu verstehen. Zwischen den Ordnungsbereichen sowie zwischen deren Funktionen bestehen Austauschbeziehungen und Wechselwirkungen. Der Ordnungsrahmen für das Event-Management betont durch sein Design die Gleichwertigkeit der Teilprozesse zum Management von Events. Die Analogien der Ordnungsbereiche sind durch deren parallele Anordnung gekennzeichnet. Des Weiteren betont die Anordnung der Phasen „Event-Planung“ und „Event-Realisierung“ deren enge zeitliche Verknüpfung.

Die Verbindung der separierten Prozesse im Sinne der Herstellung eines Ganzen ist ebenso durch die Anordnung der einzelnen Ordnungsbereiche versinnbildlicht. Auf diese Weise wird repräsentiert, dass die Planung, Durchführung und Kontrolle von Events in einer überaus engen interdependenten Beziehung stehen. Eine besondere Rolle kommt dem Ordnungsbereich der Event-Kontrolle zu. Dieser berücksichtigt Funktionen, deren Verrichtung permanent erfolgt, die der Evolution der zu erzeugenden Leistungen dienen und die sowohl die Planung als auch die Durchführung von Events unterstützen. Die Event-Kontrolle ist daher in der Mitte angeordnet. Das Projektmanagement stellt einen übergreifenden Aspekt dar und bildet somit ein Fundament für die Durchführung jeder Art von Events entlang aller Phasen des Event-Managements [Tool00].

Die besondere Zielsetzung von Projekten, welche der Planung eines Events dienen, besteht darin, im Ergebnis ein „Modell" hervorzubringen, das in der Folge zur Unterstützung der Durchführung eines Events herangezogen wird. Aufgrund der personellen und zeitlichen Separation des Planungs- und Durchführungsprozesses, ist ein Event-Manager angehalten, die Anforderungen der potenziellen Event-Teilnehmer in frühen Planungsphasen zu prognostizieren. Diese unsicherheitsbehafteten Prognosen sind aufgrund ihres starken Einflusses auf die Wirtschaftlichkeit des Vorgehens in Event-Management-Projekten wiederholten Änderungen unterworfen. Das Vorgehen in Event-Management-Projekten, für das der Ordnungsrahmen eine Empfehlung ausspricht, ist daher nicht rein sequenziell zu vollziehen. Rücksprünge in vorgelagerte Phasen sind jederzeit möglich. Auf diese Weise wird gewährleistet, dass beispielsweise in frühen Phasen definierte Anforderungen an das Event später widerrufen oder korrigiert werden können.

Der Ordnungsrahmen spricht mit Hilfe der durch ihn identifizierten Bereiche und Funktionen eine Empfehlung für das Vorgehen in Projekten aus, in denen Events geplant und/oder realisiert werden. Da dieses Vorgehen in der Praxis variiert, ist der Ordnungsrahmen projektspezifisch anzupassen. Beispielsweise würde sich im Falle der Auftragsplanung eines Events, in welcher ein Kunde klare Vorstellungen über die zu erreichende Zielgruppe des Events hat, der Aufwand zur Definition einer Zielgruppe erheblich reduzieren. Für den Fall, dass der Kunde die Zielgruppe bereits eindeutig identifizieren kann (z. B. bei einer unternehmensinternen Feierlichkeit), könnte die entsprechende Funktion sogar entfallen. Es wäre lediglich eine Fixierung der sich aus der Zielgruppe ergebenden spezifischen Umfeldbedingungen erforderlich. Diese Anpassung würde zu einem projektspezifischen Ordnungsrahmen führen, zu dessen Konstruktion der Ordnungsrahmen für das Event-Management genutzt worden wäre. Letzterer wäre damit aufgrund seiner Wiederverwendung selbst als Referenzmodell gekennzeichnet.

In den nachfolgenden Abschnitten werden die Bereiche des Ordnungsrahmens für das Vorgehen zum Management von Events erläutert. Jedem Ordnungsbereich ist jeweils ein Abschnitt gewidmet.[1] Die Ordnungsbereiche werden nacheinander behandelt, wobei aber bei der Erläuterung auf bereits behandelte analoge Tatbestände hingewiesen wird,

[1] Events haben Projektcharakter und benötigen ein eigenes Management zur zielgerichteten und professionellen Konzeption, Organisation und Durchführung. Eine Betrachtung der Methoden und Konzepte des Projektmanagements ist daher hilfreich für die erfolgreiche Durchführung eines Events. Da diese Aspekte jedoch keinen eventspezifischen Inhalt aufweisen, wird auf die Darstellung eines entsprechenden Prozessmodells verzichtet.

um Redundanzen in der Darstellung zu vermeiden. Die Ausführungen spezifizieren, neben den durch die Funktionen ausgesprochenen Handlungsempfehlungen, zusätzliche Anforderungen an die Funktionalitäten eines Informationssystems, welches Event-Manager bei der Erfüllung ihrer Aufgaben unterstützen soll.

6.2 Konstruktion der Detailmodelle

6.2.1 Event-Strategie

Eine Strategie beschreibt eine genau geplante Vorgehensweise für ein Vorhaben, d. h. sie dient als Grundlage für weitere Planungen. Eine vollständige, strategische Vorarbeit ist von zentraler Bedeutung für das Event-Management. Das EPK-Referenzmodell der Event-Strategie ist in Abbildung 3 dargestellt. Die beiden durch das Modell repräsentierten Startereignisse verdeutlichen die Tatsache, dass der Prozess des Event-Managements innerhalb eines Unternehmens beginnen oder von Kunden an einen entsprechenden Dienstleister, z. B. eine Event-Agentur, herangetragen werden kann.

Im Rahmen einer umfassenden Situationsanalyse werden Ziele und Zielgruppen des Events definiert. Um die spätere Evaluation des Events zu ermöglichen, ist es erforderlich, die Messbarkeit der Ziele zu gewährleisten. Dazu kann zunächst eine Einteilung der Ziele in strategische und operative Ziele vorgenommen werden. Ökonomische Ziele werden formuliert, um den finanziellen Erfolg messbar zu machen. Sie können neben direkt veranstaltungsrelevanten Einnahmen auch Umsatzsteigerung, Marktanteilssteigerung oder Steigerung der Kaufintensität umfassen. Kontaktziele können beispielsweise durch die Anzahl der Anmeldungen oder die Teilnehmerzahl operationalisiert werden.

Event-Ziele sind über das Event-Marketing mit der Kommunikationspolitik eines Unternehmens und somit direkt mit der übergeordneten Unternehmensstrategie verbunden. Die Ableitung der Zielstruktur eines Events muss mit den Vorgaben der Unternehmensstrategie abgeglichen werden. Im Falle von Unstimmigkeiten, müssen diese überarbeitet werden.

Eng verbunden mit der Zieldefinition ist die Eingrenzung der Zielgruppe, um Streuverluste so gering wie möglich zu halten. Für Events werden im Allgemeinen Primär- und Sekundärzielgruppen definiert (vgl. Abbildung 3). Unter der Primärzielgruppe versteht man alle Personengruppen, die unmittelbar an einem Event teilnehmen. Die Sekundärzielgruppe wird über Medien oder andere Kommunikationsformen in das Event integriert. Meist besteht die Sekundärzielgruppe aus der nicht direkt an dem Event partizipierenden Öffentlichkeit. Innerhalb der Funktion „Primärzielgruppe konkretisieren“ werden zusätzliche Informationen gesammelt. Daraus lassen sich anschließend die Struktur der Zielgruppe sowie Erfahrungswerte über die Zielgruppe ableiten. Die Definition der Zielgruppenstruktur geht über die Erfassung von Alter, Wohnort und Kaufkraft hinaus. Vielmehr müssen differenziertere Verfahren, wie etwa Lifestyle-Gruppierungen oder Szenen-Marketing, herangezogen werden. Die detaillierte Kenntnis der Zielgruppenstruktur gewährleistet einen hohen Grad an Individualität und somit eine hohe Kontaktintensität.

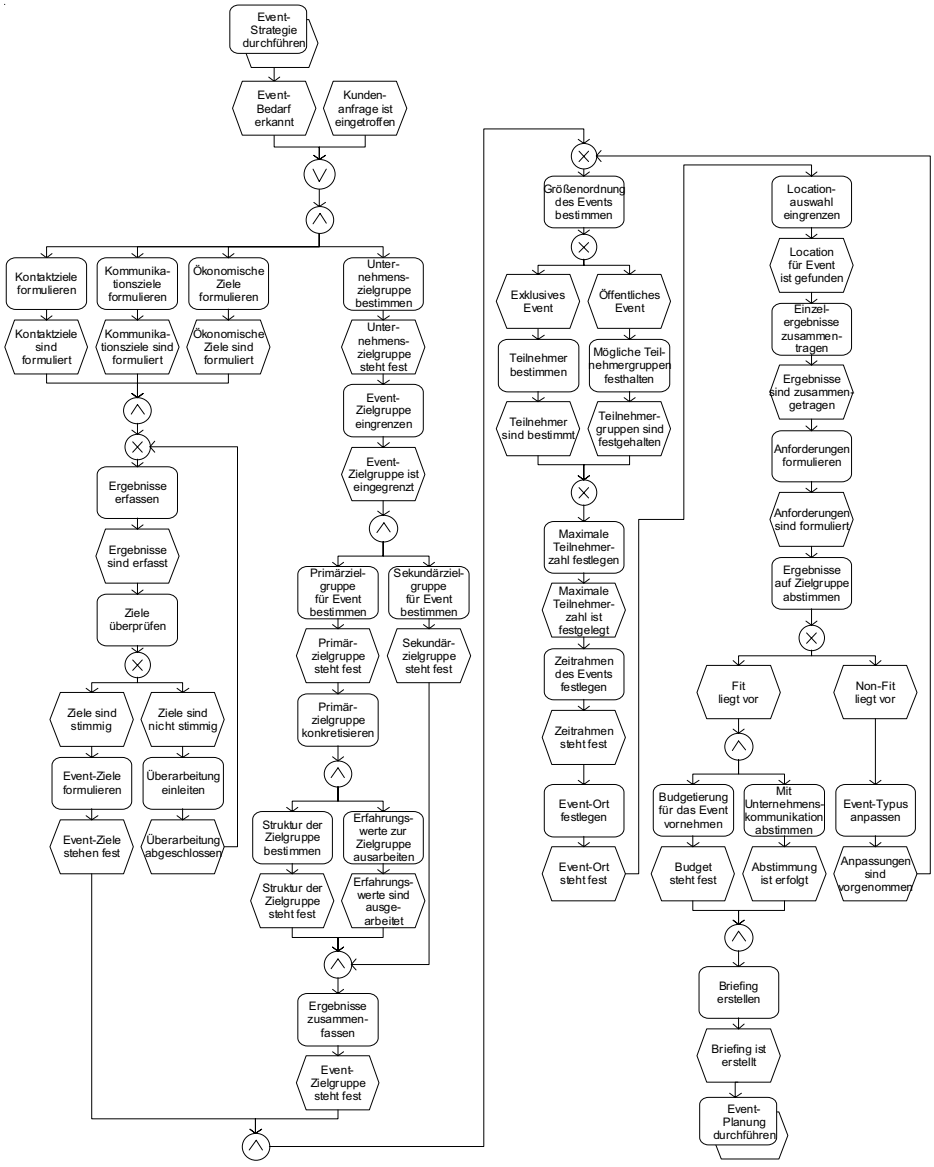

Abbildung 3: Referenzprozessmodell der Event-Strategie

An die Definition der Ziele und Zielgruppen des Events schließt sich die Konkretisierung des Event-Typus und der Rahmenbedingungen für das Event an (vgl. Abbildung 3). Zunächst wird die Größenordnung des Events bestimmt. Es folgt die Entscheidung über Exklusivität oder Öffentlichkeit des Events. Fällt die Entscheidung für ein exklusives Event, so müssen anschließend die Teilnehmer des Events bestimmt werden. Bei öffentlichen Events kann die Teilnehmerzahl stärker schwanken als bei exklusiven Events.

Daher müssen bei öffentlichen Events alle in Frage kommenden Teilnehmergruppen ermittelt werden. Nach dieser ersten Eingrenzung ist die maximale Teilnehmerzahl festzulegen. An dieser werden alle folgenden Planungen, wie etwa die Auswahl der Location oder des Catering, ausgerichtet. Im Anschluss daran muss der genaue Zeitrahmen des Events bestimmt werden, wodurch eine erste Terminfestlegung erfolgt. Events können eintägig (z. B. Gala-Abend oder Jubiläum), mehrtägig (z. B. Olympiade oder Tagungen) oder in Zyklen (z. B. Konzerte oder Shows) veranstaltet werden. Auf diesen Eckdaten aufbauend kann der Ort festgelegt werden. Während unter einer Location beispielsweise eine Konzerthalle verstanden wird, bezeichnet der Ort den geografischen Raum, in dem ein Event stattfindet, also beispielsweise „Stadt Frankfurt und Umgebung".

Die Einzelergebnisse für Größenordnung, Zeitrahmen und Ort werden zusammengefasst (vgl. Abbildung 3). Auf Basis dieser Daten werden Anforderungen an das Event gestellt, die als Grundlage für die weitere Planung dienen. Ein Vergleich dieser Daten mit Zielen und Zielgruppen des Events soll die Konsistenz der geplanten Veranstaltung sichern. Besteht ein „Non-Fit" (z. B. für ein Jubiläum mit hochrangigen Geschäftsführern eines Unternehmens wurde eine Turnhalle als Location gewählt), so muss der Prozess der Festlegung des Event-Typus sowie der Rahmenbedingungen erneut durchlaufen werden, um eine Anpassung zu erreichen (vgl. Schleife in Abbildung 3). Im Falle eines „Fit" ist der Teilprozess abgeschlossen. Die innerhalb der Event-Strategie-Phase erarbeiteten Ergebnisse werden nach einer abschließenden Abstimmung mit übergeordneten strategischen Vorgaben sowie einer ersten Budgetierung für das Event in einem Briefing festgehalten.

6.2.2 Event-Planung

In der Planungsphase, die in Abbildung 4 dargestellt ist, werden die innerhalb der Strategie ausgearbeiteten Vorgaben konkretisiert. Sie ist durch Prüfaktivitäten geprägt, die sich einerseits auf die Abstimmung der Vorgehensweise mit den Wünschen des Kunden beziehen und andererseits auf immer wiederkehrende bzw. ständig stattfindende kontrollierende Aktivitäten, wie die Budgetprüfung oder die Überwachung von Deadlines. Die meisten anpassenden Maßnahmen sind daher bereits in dieser Phase vorzunehmen.

Im Mittelpunkt des EPK-Modells in Abbildung 4 steht die Entwicklung des Event-Konzepts, aus dem sich die Planungsgrundlagen für das Event ableiten. Dieses gleicht einem Drehbuch mit genauer Planung jedes Zeitabschnitts. Die Event-Planung kann zunächst in die drei Unterfunktionen „Grobplanung", „Feinplanung" und „Detailplanung" gegliedert werden. Innerhalb der Grobplanung werden zunächst organisatorische Fragen geklärt. Da Events Projektcharakter besitzen, ist zunächst die Zusammensetzung des entsprechenden Projektteams festzulegen. Hier können Methoden des Projekt-Managements eingesetzt werden. Die Funktion „Team zusammenstellen" umfasst dabei sowohl die Auswahl des Projektleiters als auch der einzelnen Mitglieder. Besonders bei großen Projekten wird aus Komplexitätsgründen die Gesamtaufgabe in Teile gegliedert. Diese werden an die einzelnen Teammitglieder vergeben. Die Mitglieder des Teams werden über ihre Aufgaben und den (Kunden-) Auftrag informiert. Alle Bereiche des Unternehmens bzw. der Agentur, die in die weitere Planung des Events involviert sind, werden über den Auftrag in Kenntnis gesetzt, um ein effektives Zusammenspiel zu ermöglichen.

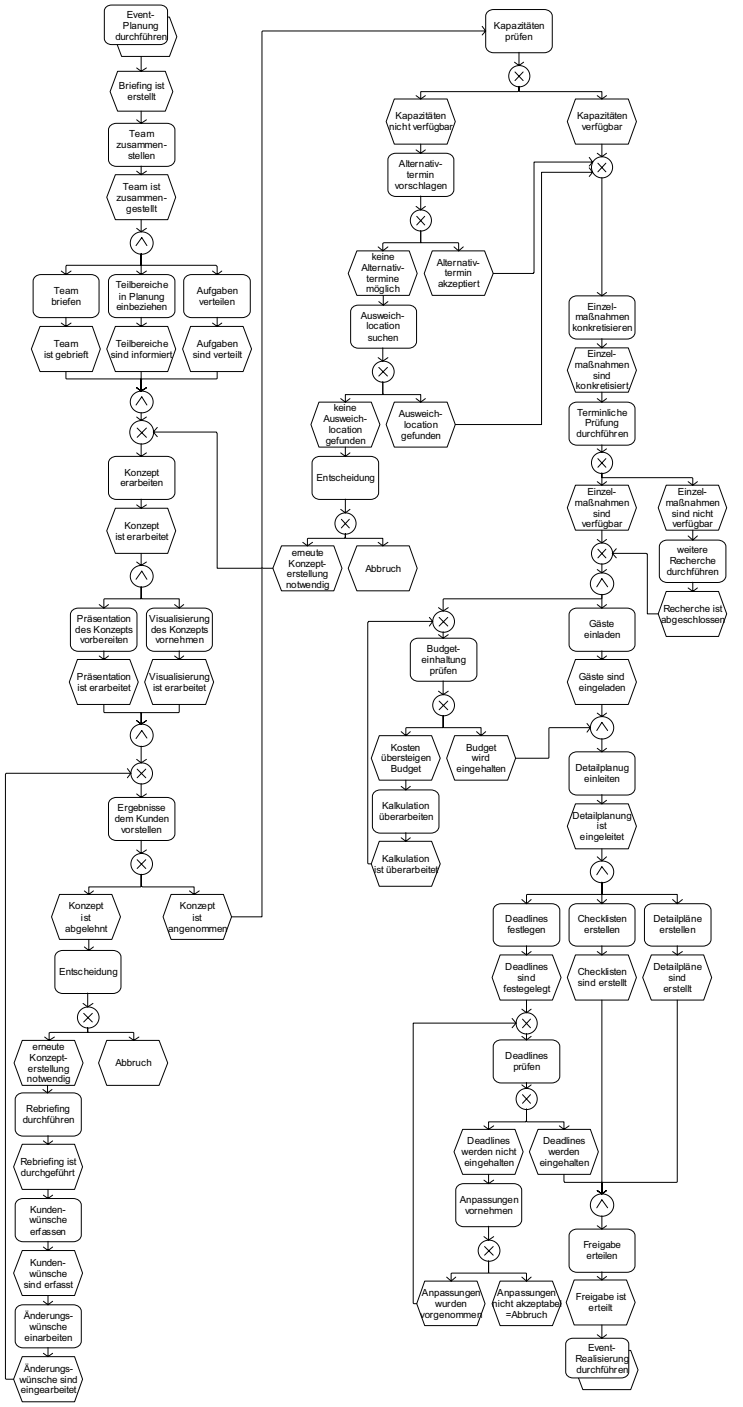

Abbildung 4: Referenzprozessmodell der Event-Planung

Nachdem das Team zusammengestellt ist und alle Aufgaben verteilt sind, geht man über in den konzeptionellen Aufgabenbereich (vgl. Abbildung 4). Hier werden erste Vorgaben und Ideen bezüglich Ort, Hotelkapazitäten bzw. Künstler auf ihre grundsätzliche Durchführbarkeit geprüft. Aufbauend auf diesen Informationen wird das Konzept entwickelt. Ein durchdachtes Konzept ist eine notwendige Voraussetzung für ein erfolgreiches Event und Ausgangspunkt für folgende Planungsschritte. Kreativitätstechniken, wie beispielsweise Brainstorming oder Mind Mapping, können hier einen Beitrag zur Ideenfindung leisten. Die Visualisierung des erdachten Konzepts hilft, den Ablauf und die Inhalte des Events greifbar zu machen. So werden in der Praxis beispielsweise dreidimensionale Präsentationen einer Location inklusive einer detaillierten Darstellung der Bühnenbilder, Dekorationen und Lichteffekte erstellt, um deren Darbietung besser einschätzen zu können. Dies ist bei der anschließenden Präsentation der Ergebnisse von Vorteil.

Eine Präsentation hat vor allem zum Ziel, den Auftraggeber von dem erstellten Konzept zu überzeugen, Änderungswünsche aufzunehmen und mögliche Unklarheiten zu beseitigen. Ist der Kunde mit den Ergebnissen zufrieden, wird die Phase der Feinplanung eingeleitet. Ist er nicht zufrieden, so muss zunächst ein Rebriefing gehalten werden (vgl. Abbildung 4). In diesem werden alle problematischen Punkte erfasst und neue Aufgabenstellungen erarbeitet. Hierzu empfiehlt es sich von Unternehmensseite her, ein Briefing-Papier als Grundlage der Diskussionen zu erarbeiten. Ein Fragenkatalog hilft dem Auftragnehmer, letzte Unklarheiten zu beseitigen. So werden die Änderungswünsche erfasst und anschließend in das Konzept eingearbeitet. Das Ergebnis dieses Überarbeitungsprozesses wird dem Kunden wiederum vorgelegt. Erst wenn dieser zustimmt, kann mit der Feinplanung begonnen werden.

Bereits in dieser frühen Phase der Planung werden die Kapazitäten der wichtigsten Veranstaltungsorte ebenso wie die der geplanten Unterkünfte oder Verkehrsmittel überprüft. Dabei ist es besonders wichtig, die Anforderungen des Kunden in diese Recherche mit einzubeziehen. Entsprechend der Kapazitätsprüfung werden anschließend die für das Event benötigten Einzelmaßnahmen konkretisiert (vgl. Abbildung 4). Diese Funktion beinhaltet die Planung der für das Event benötigten Subunternehmer, wie Catering, Künstler, Bühnentechniker usw. Anschließend werden die Einzelmaßnahmen auf ihre terminliche Verfügbarkeit hin geprüft. Für den Fall, dass bestimmte Einzelmaßnahmen nicht zum benötigten Zeitpunkt verfügbar sind, muss eine weitere Recherche durchgeführt werden.

Anschließend erfolgt die Einladung der Gäste. Dies kann bei geschlossenen Events durch das Versenden der Einladungen geschehen, bei öffentlichen Events handelt es sich dabei um die Bekanntmachung des Events sowie der dazugehörigen Termine, z.B. durch Print-, Hörfunk- oder TV-Werbung.

Bei der Prüfung des Budgets werden die endgültigen Kosten der Einzelmaßnahmen integriert, da diese vorher noch nicht zur Verfügung standen. Übersteigen die Kosten das Budget muss die Kalkulation überarbeitet werden. Anschließend wird das Budget einer weiteren Prüfung unterzogen. Sind alle Kosten innerhalb des vorgegebenen Rahmens, wird der Prozess der Detailplanung eingeleitet.

Die Detailplanung weist bereits eine große zeitliche Nähe zum eigentlichen Event auf. Auch in dieser Phase bieten Methoden des Projektmanagements, vor allem Zeitplantechniken wie Gantt-Charts, gute Unterstützungshilfen. Zunächst werden für alle Einzelmaßnahmen und noch unerledigte Aufgaben Deadlines festgelegt. Diese sollen eine zeitgerechte Bearbeitung der einzelnen Tätigkeiten gewährleisten. Jede Deadline unterliegt einer ständigen Prüfung. Kann eine Deadline nicht eingehalten werden, so sind entsprechende Anpassungen vorzunehmen. Sind diese Anpassungen inakzeptabel, so kann es zu einem Abbruch der entsprechenden Planungen kommen. Bei einer Anpassung der Deadlines unterliegen sie auch weiterhin ständiger Überwachung.

Parallel zur Erstellung der Deadlines werden in Checklisten alle bereits ausgeführten bzw. noch auszuführenden Aufgaben festgehalten. Diese Checklisten enthalten beispielsweise Joblisten für jeden Subunternehmer und deren entsprechende Aufgaben.

Neben den Checklisten müssen auch ausführliche Pläne erstellt werden. Diese Pläne sind das Ergebnis logistischer Feinarbeit und erfassen lückenlos alle organisatorischen Einzelheiten für das Event. Eine Prüfung der Pläne findet während der Realisierungsphase des Events statt. Sind alle Pläne und Checklisten erstellt und Deadlines eingehalten, kann die Freigabe für das Event erteilt werden, und die Phase der Realisierung beginnt.

6.2.3 Event-Realisierung

Unter der Realisierung eines Events wird die tatsächliche Umsetzung des Events verstanden. Die Realisierungsphase umfasst in der Regel den Zeitraum des Events, der vor Ort in direktem Kontakt mit der Zielgruppe erfolgt. Daher stehen in diesem Zeitabschnitt Prozessabläufe, Koordination der Mitwirkenden und kurzfristige Problembearbeitung im Vordergrund. Die Event-Realisierung kann in die drei Unterphasen Pre-Event, Main-Event und Post-Event eingeteilt werden (vgl. Abbildung 2). Im Pre-Event werden alle vorbereitenden Maßnahmen ausgeführt. Die Phase des Main-Events bezeichnet den Ablauf und die konkrete Durchführung des Events. In der Post-Event-Phase schließlich werden Aktivitäten ausgeführt, die nach Ende der Veranstaltung durchzuführen sind.

Neben den allgemeinen vorbereitenden, organisatorischen Aktivitäten der Pre-Event-Phase, wie etwa der Anlieferung der Technik oder des Bühnenbilds, ist die Generalprobe die wichtigste Funktion (vgl. Abbildung 5). Sie dient vor allem dazu, dem Auftraggeber einen Überblick über die während des Main-Events stattfindenden Aktivitäten zu zeigen. So werden dem Kunden beispielsweise die wichtigsten „Settings“ des Events vorgeführt. Diese Settings sind z. B. Eingangs-/Begrüßungsbereich, Buffet oder Bühne. Zusätzlich werden in der Generalprobe Licht- und Bühnentechnik optimal eingestellt („Soundcheck“). Künstler und Caterer üben ihren Auftritt bzw. überprüfen Wege und Platzangebote. Falls während der Generalprobe Mängel auftreten oder der Kunde Änderungswünsche hat, erfolgt eine Nachbesserung. Sind die Ergebnisse dieser Nachbesserung zufrieden stellend, kann die Freigabe für das Main-Event erteilt werden und alle Mitwirkenden stehen für das Main-Event bereit. Im Fachjargon der Event-Agenturen wird dieser Schritt als „Stand-By-All“ bezeichnet.

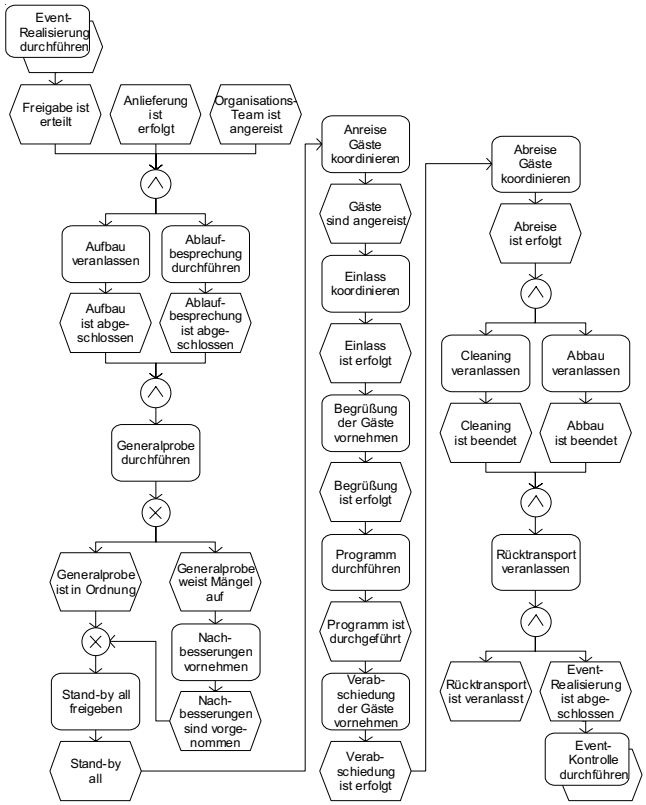

Abbildung 5: Referenzprozessmodell der Event-Realisierung

In der Main-Event-Phase findet die eigentliche Ansprache der Teilnehmer durch die Programmpunkte des Events statt. Die vorab geplanten und geprobten Abläufe müssen nun adäquat ausgeführt werden, um das Event-Ziel zu erreichen. Diese Phase der Realisierung ist je nach Art des Events sehr unterschiedlich ausgeprägt. Das Haupt-Event ist das Ergebnis des innerhalb der Planungsphase erarbeiteten Konzepts und unterliegt somit der Kreativität und des organisatorischen Geschicks des Event-Managers. Deshalb ist eine detaillierte Darstellung dieser Phase nur für den jeweiligen Einzelfall möglich. An dieser Stelle sind daher lediglich die Hauptfunktionen (Ankunft, Einlass, Begrüßung, Durchführung und Verabschiedung) des Events berücksichtigt. Die Funktion „Programm durchführen“ erfordert Aufmerksamkeit. Aufgrund seiner spezifischen und individuellen Gestaltung ist für jedes Event ein eigener spezieller Programm-Prozess zu entwickeln.

Die Verabschiedung ist zugleich der Startpunkt des Post-Events. Die Abreise der Gäste wird entsprechend der Anreise betreut. Sobald die Gäste die Location verlassen haben, kann mit Abbau und Cleaning begonnen werden. Sind diese beiden Funktionen ausgeführt, wird der Rücktransport der Materialien veranlasst. Erst wenn alle die Location betreffenden Aktivitäten erledigt sind, kann das Organisationsteam die Rückreise antreten. Die Realisationsphase ist somit abgeschlossen.

6.2.4 Event-Kontrolle

Die Kontrolle ist ein phasenübergreifender Aspekt, der zu allen Zeitpunkten stattfinden kann. Sie steuert den gesamten Event-Management-Prozess, indem Abweichungen erfasst und Ursachen für diese identifiziert werden. In den verschiedenen Phasen des Events sind unterschiedliche Kontrollmechanismen denkbar. Durch eine Machbarkeitsstudie im Vorfeld eines Events können die marktspezifischen Rahmenbedingungen ermittelt werden. Während des Events ermöglicht eine durchgängige Ablaufkontrolle und -dokumentation, Erfahrungen zu sammeln und das so generierte Wissen für weitere Veranstaltungen nutzbar zu machen. Die Ergebniskontrolle ermittelt anhand von Marktforschungsaktivitäten marketingrelevante Auswirkungen des Events, wie z. B. Imagewirkungen. Die Nachbereitung umfasst vor allem Abrechnung, abschließende Datenaufbereitung und Manöverkritik. An dieser Stelle wird lediglich der Prozess der abschließenden Kontrolle dargestellt.

Die Event-Kontrolle zeichnet sich durch die Integration von Planung, Information, Organisation und Steuerung aus. Dies bezieht sich sowohl auf die Kontrolle der Kosten, als auch auf die Überwachung des Fortschritts und der damit verbundenen Qualitätssicherung [Stic01, S. 196].

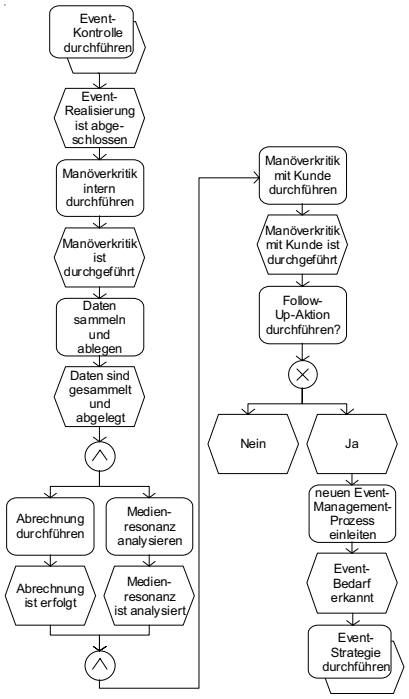

Abbildung 6: Referenzprozessmodell der Event-Kontrolle

Nachdem die Event-Realisierung abgeschlossen ist, kann die Nachbereitung durch eine abschließende Kontrolle beginnen (vgl. Abbildung 6). Zunächst wird eine interne Manö-

verkritik durchgeführt. Hierunter versteht man eine Analyse der verschiedenen Aufgabengebiete des Events im zuständigen Team, wobei beispielsweise Verbesserungsvorschläge erarbeitet werden oder besonders erfolgreiche Arbeiten festgehalten werden, um das Wissen für nachfolgende Veranstaltungen nutzbar zu machen. Anschließend werden alle das Event betreffenden Daten gesammelt. Diese Daten können aus dem vorangegangenen Feedback stammen oder während des gesamten Event-Management-Prozesses erfasst worden sein. So werden beispielsweise alle Daten der Subunternehmer in dafür vorgesehenen Datenbasen abgelegt.

Parallel zur Rechnungsstellung wird eine Analyse der Medienresonanz durchgeführt. Nur wenn eine entsprechende mediale Nachbereitung des Events erfolgt, kann die Sekundärzielgruppe zielgemäß an dem Event teilhaben. Die Ergebnisse der Medienresonanzanalyse liefern Informationen für die Manöverkritik, die abschließend mit dem Kunden durchgeführt wird. Je nach Event-Typus wird die Entscheidung gefällt, ob ein Follow-Up-Event durchgeführt wird und damit ein neuer Prozess des Event-Managements durchlaufen wird.

7 Diskussion der Ergebnisse und weiterer Forschungsbedarf

Gegenstand des vorliegenden Artikels war die Konstruktion eines Referenzprozessmodells für das Event-Management. Das Referenzmodell spricht Empfehlungen zur Gestaltung prozessorientierter Informationssysteme aus, die der Unterstützung des Event-Managements dienen. Die hiermit beabsichtigte Verbindung zweier sich getrennt voneinander entwickelnder Forschungsgebiete – einerseits Event-Marketing bzw. Event-Management als Teildisziplin der Betriebswirtschaftslehre und andererseits Referenzmodellierung als Teildisziplin der Wirtschaftsinformatik – ist aus zweierlei Hinsicht neuartig: Einerseits liegen bislang keine nennenswerten Forschungsergebnisse zur Modellierung von Event-Management-Systemen vor. Andererseits kann mit den Konstruktionsergebnissen dieses Beitrags auf den im Arbeitsgebiet der Referenzmodellierung vielfach kritisierten Mangel an wiederverwendbaren Domänenmodellen reagiert werden.

Die Konstruktion des Referenzmodells wurde – wie in der Referenzmodellierung üblich – in die Erstellung eines Ordnungsrahmens, das Event-E, und die Modellierung der den Bereichen dieses Ordnungsrahmens zugeordneten Detailmodelle untergliedert. Während die Konstruktion der Detailmodelle mit der EPK auf einer etablierten domänenneutralen Prozessbeschreibungssprache aufbaute, zeigten die Begründungen zum Aufbau des Event-E, dass bei der Konstruktion von Referenzmodellordnungsrahmen Überlegungen dominieren, die Symbolcharakter besitzen. Hiermit soll beim Betrachter eine Versinnbildlichung der durch das Modell beschriebenen Zusammenhänge erreicht werden. Die Gedankengänge sind auf die jeweiligen Gegenstandsbereiche ausgerichtet und daher kaum mit anwendungsdomänenneutralen Modellierungssprachen zum Ausdruck zu bringen. Neben der mit ihnen beabsichtigten Modellstrukturierung dienen Ordnungsrahmen somit auch als Markenzeichen der ihnen zugeordneten Referenzmodelle.

An diesem Punkt setzt jedoch häufig die Kritik an diesen Ordnungsrahmen an. So gibt beispielsweise BECKER zu bedenken, dass Referenzmodellordnungsrahmen im Sinne e-

lementarer grafischer Darstellungen auf hohem Abstraktionsniveau „der Gefahr ausgesetzt [sind], als ‚ikonenhaft' und wenig wissenschaftlich abgetan zu werden" [Beck96, S. 16]. SCHEER [Sche00, S. 145f.] sieht jedoch gerade in der Repräsentation komplexer Anwendungsdomänen durch einfache grafische Symbole mit Logocharakter auch zukünftig eine wissenschaftliche Herausforderung für die Wirtschaftsinformatik.

Danksagung. Dieser Beitrag resultiert aus dem Forschungsprojekt „Referenzmodell-gestütztes Customizing unter Berücksichtigung unscharfer Daten", Kennwort: Fuzzy-Customizing, Teilprojekt der Forschungskohorte „Betriebliche Referenz-Informationsmodellierung – Designtechniken und domänenbezogene Anwendung" (BRID^2), gefördert von der Deutschen Forschungsgemeinschaft (Förderkennzeichen: SCHE 185/25–1).

Literatur

[BDKK02] Becker, J. et al.: Konfigurative Referenzmodellierung. In: Becker, J.; Knackstedt, R. (Hrsg.): *Wissensmanagement mit Referenzmodellen : Konzepte für die Anwendungssystem- und Organisationsgestaltung.* Heidelberg [u. a.] : Physica, 2002, S. 25–144

[Beck96] Becker, J.: Eine Architektur für Handelsinformationssysteme. In: Becker, J. et al. (Hrsg.): *Arbeitsberichte des Instituts für Wirtschaftsinformatik*, Nr. 46, Münster : Westfälische Wilhelms-Universität, 1996

[BeSc04] Becker, J.; Schütte, R.: *Handelsinformationssysteme : Domänenorientierte Einführung in die Wirtschaftsinformatik.* 2. vollst. überarb., erw. u. aktual. Aufl. Landsberg/Lech : Moderne Industrie, 2004

[Clar04] Clarke, A.: Evaluating Mega-Events : A Critical Review. In: *Proceedings of the 3rd DeHaan Tourism Management Conference "The Impact and Management of Tourism-Related Events" ; December 14th 2004 , University of Nottingham*, 2004

[CuKL98] Curran, T. A.; Keller, G.; Ladd, A.: *SAP R/3 business blueprint : Understanding the business process reference model.* Upper Saddle River, NJ : Prentice Hall PTR, 1998

[Dren03] Drengner, J.: *Imagewirkungen von Eventmarketing : Entwicklung eines ganzheitlichen Messansatzes.* Wiesbaden : Dt. Univ.-Verlag, 2003. – Zugl.: Chemnitz, Techn. Univ., Diss., 2003

[Erbe02] Erber, S.: *Eventmarketing : Erlebnisstrategien für Marken ; Innovative Konzepte, zahlreiche Fallbeispiele, viele Tipps zur Umsetzung in der Praxis.* 3. Aufl. München : Redline Wirtschaft bei Verl. Moderne Industrie, 2002

[FeLo03] Fettke, P.; Loos, P.: Classification of Reference Models – A Methodology and its Application. In: *Information Systems and e-Business Management* 1 (2003), Nr. 1, S. 35–53

[Fres00] Frese, E.: *Grundlagen der Organisation : Konzept – Prinzipien – Strukturen.* 8., überarb. Aufl. Wiesbaden : Gabler, 2000

[Geib97] Geib, T.: *Geschäftsprozeßorientiertes Werkzeugmanagement.* Wiesbaden : Gabler, 1997. – Zugl.: Saarbrücken, Univ., Diss.

[Getz97] Getz, D.: *Event Management & Event Tourism.* New York : Cognizant Communication Corporation, 1997

[GGGP71] Grochla, E. et al.: Grundmodell zur Gestaltung eines integrierten Datenverarbeitungssystems : Kölner Integrationsmodell (KIM). In: Grochla, E.; Szyperski, N. (Hrsg.): *Arbeitsberichte des Betriebswirtschaftlichen Instituts für Organisation und Automation (BIFOA) an der Universität zu Köln*, Nr. 71/6, Köln : WISON Verl., 1971

[Gold00] Goldblatt, J. J.: A future for Event Management : The analysis of major trends impacting the emerging profession. In: Allen, J. et al. (Hrsg.): *Events beyond 2000 : Setting the Agenda ; Proceedings of Conference on Event Evaluation, Research and Education ; July 2000 , Sydney.* Sydney : Australian Centre for Event Management, 2000, S. 1–9

[GPJC02] The George P Johnson Company (Hrsg.): *Trends in U.S. Marketing 2001–2002.* Detroit, 2002

[Habe92] Haberfellner, R.: Projektmanagement. In: Frese, E. (Hrsg.): *Handwörterbuch der Or-*

ganisation. 3., völlig neu gest. Aufl. Stuttgart : Poeschel, 1992, S. 2090–2102

[Hay03] Hay, D. C.: *Requirements analysis : From business views to architecture.* Upper Saddle River, NJ : Prentice Hall PTR, 2003

[HeJD02] Hede, A.-M.; Jago, L. K.; Deery, M.: Special Event Research 1990–2001 : Key Trends and Issues. In: Australian Center for Event Management (Hrsg.): *Events & Place Making : Event Research Conference ; 15–16 July 2002 , Sydney.* University of Technology Sydney, 2002, S. 305–338

[Hol+03] Holzbaur, U. et al.: *Eventmanagement : Veranstaltungen professionell zum Erfolg führen.* 2., erw. Aufl. Berlin [u. a.] : Springer, 2003

[IDS03a] IDS Scheer AG (Hrsg.): *ARIS Methode : November 2003.* Saarbrücken : IDS Scheer AG, 2003

[IDS03b] IDS Scheer AG (Hrsg.): *Meilensteine in der Unternehmensgeschichte*. URL http://www.ids-scheer.de/international/german/investor/milestones [Zugriffsdatum 01.05.2003]

[JaSh98] Jago, L. K.; Shaw, R. N.: Special events : A conceptual and differential framework. In: *Festival Management and Event Tourism* 5 (1998), Nr. 1/2, S. 21–32

[KeNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". In: Scheer, A.-W. (Hrsg.): *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, Nr. 89, Saarbrücken : Universität des Saarlandes, 1992

[KeTe99] Keller, G.; Teufel, T.: *SAP R/3 prozeßorientiert anwenden : Iteratives Prozess-Prototyping mit Ereignisgesteuerten Prozessketten und Knowledge Maps.* 3., erw. Aufl. Bonn [u. a.] : Addison-Wesley, 1999

[Kilo02] Kilov, H.: *Business models : A guide for business and IT.* Upper Saddle River : Prentice Hall, 2002

[Krus96] Kruse, C.: *Referenzmodellgestütztes Geschäftsprozeßmanagement : Ein Ansatz zur prozeßorientierten Gestaltung vertriebslogistischer Systeme.* Wiesbaden : Gabler, 1996. – Zugl.: Saarbrücken, Univ., Diss., 1995, u.d.T.: Geschäftsprozeßmanagement in vertriebslogistischen Systemen

[Lang97] Lang, K.: *Gestaltung von Geschäftsprozessen mit Referenzprozeßbausteinen.* Wiesbaden : DUV [u. a.], 1997. – Zugl.: Erlangen, Nürnberg, Univ., Diss., 1996

[Lars03] Larson, M.: *Evenemangsmarknadsföringens organisering : Interaktion mellan aktörer på ett politiskt torg.* Göteborg : Handelshögskolan, Universitet, 2003 (Vetenskapliga bokserien / ETOUR; 2003,11). – Zugl.: Göteborg, Handelshögskolan vid Göteborgs Univ., Diss. ; Zusammenfassung in engl. Sprache u.d.T.: Organising the marketing of events

[Lass03] Lasslop, I.: *Effektivität und Effizienz von Marketing-Events : Wirkungstheoretische Analyse und empirische Befunde.* Wiesbaden : Gabler, 2003. – Zugl.: Münster, Univ., Diss.

[LaTB96] Lang, K.; Taumann, W.; Bodendorf, F.: Business Process Reengineering with reusable Reference Process Building Blocks. In: Scholz-Reiter, B.; Stickel, E. (Hrsg.): *Business process modelling.* Berlin [u. a.] : Springer, 1996, S. 265–290

[Litk04] Litke, H.-D.: *Projektmanagement : Methoden, Techniken, Verhaltensweisen ; Evolutionäres Projektmanagement.* 4., überarb. und erw. Aufl. München [u. a.] : Hanser, 2004

[Lupp04] Luppold, S.: EDV in der Veranstaltung. In: Haase, F.; Mäcken, W. (Hrsg.): *Handbuch Event-Management.* München : kopaed, 2004, S. 129–143

[Mada00] Madauss, B. J.: *Handbuch Projektmanagement : mit Handlungsanleitungen für Industriebetriebe, Unternehmensberater und Behörden.* 6., überarb. u. erw. Aufl. Stuttgart : Schäffer-Poeschel, 2000

[Meis01] Meise, V.: *Ordnungsrahmen zur prozessorientierten Organisationsgestaltung : Modelle für das Management komplexer Reorganisationsprojekte.* Hamburg : Kovac, 2001. – Zugl.: Münster (Westfalen), Univ., Diss., 2000

[Müll03] Müller, W.: *Eventmarketing : Grundlagen, Rahmenbedingungen, Konzepte, Zielgruppe, Zukunft.* 2., überarb. Aufl. Düsseldorf : VDM-Verl. Müller, 2003

[Mylo98] Mylopoulos, J.: Information Modeling in the Time of the Revolution. In: *Information Systems* 23 (1998), Nr. 3/4, S. 127–155

[Nufe02] Nufer, G.: *Wirkung von Event-Marketing : Theoretische Fundierung und empirische Analyse.* Wiesbaden : Dt. Univ. Verl., 2002. – Zugl.: Tübingen, Univ., Diss., 2001

[NüRu02] Nüttgens, M.; Rump, F. J.: Syntax und Semantik Ereignisgesteuerter Prozessketten

(EPK). In: Desel, J.; Weske, M. (Hrsg.): *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (Promise '2002) : Gemeinsames Fachgruppentreffen der GI-Fachgruppen "Petrinetze und verwandte Systemmodelle" und "Entwicklungsmethoden für Informationssysteme und ihre Anwendung" (EMISA), Hasso-Plattner-Institut für Softwaresystemtechnik an der Universität Potsdam, 9.–11. Oktober 2002.* Bonn : Köllen, 2002, S. 64–77

[Nütt95] Nüttgens, M.: *Koordiniert-dezentrales Informationsmanagement : Rahmenkonzept – Koordinationsmodelle – Werkzeug-Shell.* Wiesbaden : Gabler, 1995. – Zugl.: Saarbrücken, Univ., Diss.

[Pepe98] Pepels, W.: *Marketing : Lehr- und Handbuch mit Praxisbeispielen.* 2., bearb. u. erw. Aufl. München [u. a.] : Oldenbourg, 1998

[Remm97] Remme, M.: *Konstruktion von Geschäftsprozessen : Ein modellgestützter Ansatz durch Montage generischer Prozeßpartikel.* Wiesbaden : Gabler, 1997. – Zugl.: Saarbrücken, Univ., Diss., 1996, u.d.T.: Geschäftsprozeßkonstruktion durch Montage generischer Prozeßpartikel

[RoSi01] Rossi, M.; Siau, K.: *Information Modeling in the new Millennium.* Hershey [u. a.] : Idea Group Publishing, 2001

[ScAT02] Scheer, A.-W.; Angeli, R.; Thomas, O.: eLogistics : Kundenorientierte Planung und Steuerung von Güter- und Informationsflüssen in Unternehmensnetzwerken. In: Manschwetus, U.; Rumler, A. (Hrsg.): *Strategisches Internetmarketing : Entwicklungen in der Net-Economy.* Wiesbaden : Gabler, 2002, S. 457–480

[Sche00] Scheer, A.-W.: *Unternehmen gründen ist nicht schwer ...* Berlin [u. a.] : Springer, 2000

[Sche97] Scheer, A.-W.: *Wirtschaftsinformatik : Referenzmodelle für industrielle Geschäftsprozesse.* 7., durchges. Aufl. Berlin [u. a.] : Springer, 1997

[Sche99] Scheer, A.-W.: *ARIS – business process modeling.* 2., completely rev. and enl. ed. Berlin [u. a.] : Springer, 1999

[Schü98] Schütte, R.: *Grundsätze ordnungsmäßiger Referenzmodellierung : Konstruktion konfigurations- und anpassungsorientierter Modelle.* Wiesbaden : Gabler, 1998. – Zugl.: Münster (Westfalen), Univ., Diss., 1997

[Schw04] Schwandner, G.: Grundlagen – Projektmanagement und Organisation. In: Haase, F.; Mäcken, W. (Hrsg.): *Handbuch Event-Management.* München : kopaed, 2004, S. 27–45

[Schw99] Schwegmann, A.: *Objektorientierte Referenzmodellierung : Theoretische Grundlagen und praktische Anwendung.* Wiesbaden : Gabler, 1999. – Zugl.: Münster (Westfalen), Univ., Diss.

[ScTh05] Scheer, A.-W.; Thomas, O.: Geschäftsprozessmodellierung mit der ereignisgesteuerten Prozesskette. In: *Das Wirtschaftsstudium* 34 (2005), Nr. 8–9, S. 1069–1078

[Stic01] Stickel, E.: *Informationsmanagement.* München [u. a.] : Oldenbourg, 2001

[Thom05] Thomas, O.: Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation. In: Kindler, E.; Nüttgens, M. (Hrsg.): *Business Process Reference Models : Proceedings of the Workshop on Business Process Reference Models (BPRM 2005) ; Salellite workshop of the Third International Conference on Business Process Management (BPM), Nancy, France, September 5, 2005.* Nancy, 2005, S. 16–29. – URL http://wwwcs.uni-paderborn.de/cs/kindler/events/BPRM05/PDF/BPRM05_Proceedings.pdf

[Tool00] O'Toole, W.: Towards the integration of Event Management Best Practice by the Project Management Process. In: Allen, J. et al. (Hrsg.): *Events beyond 2000 : Setting the Agenda ; Proceedings of Conference on Event Evaluation, Research and Education ; July 2000 , Sydney.* Sydney : Australian Centre for Event Management, 2000, S. 86–92

[vBro03] vom Brocke, J.: *Referenzmodellierung : Gestaltung und Verteilung von Konstruktionsprozessen.* Berlin : Logos, 2003. – Zugl.: Münster (Westfalen), Univ., Diss., 2002

[WaWe02] Wand, Y.; Weber, R.: Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda. In: *Information Systems Research* 13 (2002), Nr. 4, S. 363–376

[Webe97] Weber, R.: *Ontological foundations of information systems.* Melbourne : Coopers & Lybrand and the Accounting Association of Australia and New Zealand, 1997

[ZaDr03] Zanger, C.; Drengner, J.: Eventreport 2003. In: *Veröffentlichungen der Professur für Marketing und Handelsbetriebslehre*, Nr. 3, Technische Universität Chemnitz, 2004

Einsatz von Ereignisgesteuerten Prozessketten zur Modellierung von Prozessen in der Krankenhausdomäne – Eine empirische Methodenevaluation

Kamyar Sarshar[1], Philipp Dominitzki[2], Peter Loos[1]

[1]Institut für Wirtschaftsinformatik (IWi) im DFKI
Stuhlsatzenhausweg 3, Geb. D3 2
D-66123 Saarbrücken, Germany
E-Mail: {sarshar|loos}@iwi.uni-sb.de

[2]GIP AG Research Institute
Göttelmannstraße 17
D-55130 Mainz, Germany
E-Mail: philipp.dominitzki@gip.com

Abstract: Die vorliegende Arbeit untersucht die Anwendbarkeit der EPK und EPK-Erweiterungen zur Modellierung von Prozessen in der Krankenhausdomäne. Neben den Potentialen sollen vor allem die Grenzen der Methode aufgezeigt werden, um Anhaltspunkte für zukünftige Weiterentwicklungen und Anpassungen zu erhalten. Die Untersuchung zeigt, dass Anforderungen bezüglich der Abbildung des regelbasierten Kontrollflusses und der Organisation zu großen Teilen von der EPK zufriedenstellend abgedeckt werden können. Defizite bestehen bei der Darstellung der Zeit, der internen Zustandsdynamik des Prozessobjekts "Patient", Beschreibung unterschiedlicher Iterationsvarianten und der Repräsentation von nichtregelbasierten und intuitiven Entscheidungssituationen von Ärzten und Pflegern.

1 Motivation

Mit der Abschaffung des Selbstkostendeckungsprinzips sowie der schrittweisen Einführung eines leistungsorientierten Fallpauschalensystems seitens des Gesetzgebers betreten Krankenhäuser zunehmend ein betriebswirtschaftlich geprägtes Marktumfeld [Lü03]. Vor diesem Hintergrund findet derzeit in Krankenhäusern ein Umdenken von einer an medizinischen Teildisziplinen und Berufsgruppen (Ärzte, Pfleger, Verwaltung) orientierten, funktionalen Organisation hin zu einer prozessorientierten Sichtweise statt, die die stations- und berufsgruppenübergreifende Behandlung des Patienten von der Aufnahme bis zur Entlassung in den Vordergrund stellt [Br05]. Solche Prozesse werden nachfolgend als Behandlungsprozesse bezeichnet.

Während zur Modellierung betrieblicher Geschäftsprozesse eine Reihe bewährter Methoden bereitstehen, fehlen derzeit bezüglich der Beschreibung von Behandlungsprozes-

sen in der Krankenhausdomäne noch Erfahrungswerte. Ein möglicher Ansatz ist, bei der Modellierung von Behandlungsprozessen auf die aus kommerziellen Projekten vor allem im SAP-Umfeld [Ke99] und im Zusammenhang mit dem ARIS-Toolset [Sc94] bekannte Ereignisgesteuerte Prozesskette (EPK) [Ke92] zurückzugreifen. Bisherige Erfahrungen mit der EPK legen die Vermutung nahe, dass die Methode in der Lage ist, Verwaltungstätigkeiten wie die Patientenaufnahme oder die Leistungsabrechung mit Krankenkassen befriedigend zu beschreiben. Für eine berufgruppenübergreifende und ganzheitliche Beschreibung eines Behandlungsprozesses muss jedoch noch geklärt werden, inwieweit die EPK auch pflegerische und ärztliche Tätigkeiten als integrale Bestandteile eines Behandlungsprozesses darstellen kann.

Ziel der vorliegenden Arbeit ist es, die EPK und EPK-Erweiterungen auf ihre Anwendbarkeit zur Modellierung von Behandlungsprozessen in der Krankenhausdomäne hin zu evaluieren. Dabei sollen neben den Potentialen vor allem die Grenzen der Methode aufgezeigt werden, um so Anhaltspunkte für zukünftige Weiterentwicklungen und Anpassungen der EPK zu erhalten.

Der Aufbau der Arbeit gestaltet sich wie folgt: In Kapitel 2 werden verwandte Arbeiten aufgearbeitet, um den aktuellen Stand der Forschung zu skizzieren. Anschließend wird in Kapitel 3 die genutzte Evaluationsmethode beschrieben und gegenüber alternativen Evaluationsmethoden positioniert. Den Hauptteil der Arbeit bildet Kapitel 4. Hier werden die spezifischen Anforderungen der Modellierung in der Krankenhausdomäne anhand empirisch erhobener Behandlungsprozesse hergeleitet. Durch die vergleichende Betrachtung des hergeleiteten Anforderungskatalogs der Krankenhausdomäne und der Ausdrucksmächtigkeit der EPK und vorgeschlagene EPK-Erweiterungen werden Potentiale und Grenzen dieser Methode aufgezeigt. Kapitel 5 diskutiert diese Ergebnisse. Im abschließenden Kapitel 6 werden die Ergebnisse des Beitrags zusammengefasst.

2 Verwandte Arbeiten

Im Bereich der domänenspezifischen Modellierungsmethoden existieren eine Reihe dedizierter Ansätze zur Modellierung von Behandlungsprozessen [Wa02]. Die Untersuchung dieser Methoden macht allerdings deutlich, dass sie noch nicht über das prototypische Stadium hinausgekommen sind, noch wenig Erfahrung mit deren Einsatz in realen Projekten existiert und für diese bislang keine ausgereiften Modellierungswerkzeuge zur Verfügung stehen [SL04].

Tabelle 1 listet die bislang zur Darstellung von Behandlungsprozessen verwendeten domänenneutralen Modellierungsmethoden. Als erstes Ergebnis dieser Zusammenstellung kann festgehalten werden, dass bereits diverse Versuche unternommen wurden, etablierte Modellierungsmethoden in der Krankenhausdomäne anzuwenden. Jedoch ist die gesamte Anzahl der identifizierten Publikationen im Verhältnis zur Relevanz des Themas vergleichsweise gering. Weiterhin basieren diese größtenteils auf deduktiv-theoretischen Überlegungen, so dass eine empirische Überprüfung der Eignung der Methoden bislang noch aussteht. Diese Lücke soll mit der vorliegenden Arbeit für die EPK geschlossen werden.

Modellierungsmethode	Quellen
Fluss- / Blockdiagramme	• GREILING; HOFSTETTER [GH02] • DYKES [Dy02] • KÜTTNER [Kü04]
Petri-Netze	• SARSHAR; LOOS [SL05] • GROTE et al. [Gr99] • JØRGENSEN [Jø03]
Ereignisgesteuerte Prozessketten (EPK)	• SCHEER et al. [Sc96] • SLOANE; WAGNER [SW04] • GOSPODAREVSKAYA et al. [Go05] • PERREVORT [PE03] • GREILING; HOFSTETTER [GH02] • MOSA [Mo01] • MIDDENDORF [Mi01] • VON EIFF; ZIEGENBEIN |VEZ03]
Erweitertes Strategic-Rationale-Diagramm (xSRD)	• KIRN et al. [Ki00]
Erweitertes Strategic-Dependency-Diagramm (xSDD)	
Agentenorientiert-erweiterte EPK (xEPK)	
ADEPT	• DADAM; REICHERT [DR00]
WorkParty	• REICHERT et al. [Re97]

Tabelle 1: Domänenneutrale Modellierungssprachen zur Abbildung von Behandlungsprozessen

3 Methode der Evaluation

Auf welche Weise die Evaluation einer Methode zu erfolgen hat, ist Thema einer breiten Diskussion in der Literatur. Jedoch haben sich bislang keine allgemein anerkannten Theorien oder Kriterien hierzu etabliert. Um die diskutierten Evaluationsmethoden zu systematisieren, erfolgt in Tabelle 2 – in Anlehnung an [SR98] – eine Kategorisierung der Ansätze in nicht-empirische und empirische Evaluationsmethoden.

Eine verbreitete Evaluationsmethode ist die Überprüfung, inwieweit die zu untersuchende Methode in der Lage ist, charakteristische Gegebenheiten einer Domäne zu repräsentieren (vgl. z.B. [Fl86], [Su88], [Lo92, S. 44-88]). Dabei können die gestellten Anforderungen rein deduktiv oder empirisch hergeleitet und begründet werden.

Ansatz	Evaluationsmethode
Nicht-empirische Evaluation	Metamodellbasierte Evaluation
	Ontologische Evaluation
	Evaluation auf der Basis kognitiver Theorien aus der Psychologie
	Evaluation auf der Basis erwünschter Metriken
	Evaluation auf der Basis erwünschter Eigenschaften / Fähigkeiten
Empirische Evaluation	Evaluation durch Fallstudien
	Evaluation durch Aktionsforschung
	Evaluation durch Feldexperimente
	Evaluation durch Interviews und Umfragen (Surveys)
	Evaluation durch Laborexperimente / Laborsimulationen

Tabelle 2: Evaluationsmethoden in der Übersicht (in Anlehnung an [SR98])

Für die vorliegende Untersuchung wurden die zu überprüfenden Anforderungen empirisch hergeleitet. Da jedoch keine zuverlässigen empirischen Daten von Behandlungsprozessen vorlagen, die zur Formulierung von Anforderungen hätten ausgewertet werden können, wurden diese zunächst im Rahmen einer Vorstudie erhoben. Ziel dieser Vorstudie, die in Zusammenarbeit mit dem Universitätsklinikum der Johannes Gutenberg-Universität Mainz durchgeführt wurde, war die bereichs- und berufsgruppenübergreifende Erfassung von Behandlungsprozessen von der Aufnahme bis zu Entlassung von Patienten.

Zur Erhebung der Behandlungsprozesse wurden drei empirische Methoden zur Datenerhebung kombiniert:

1. 20 Patienten wurden während ihres Klinikaufenthaltes durchgehend begleitet, um den Behandlungsprozess durch eine *strukturierte Beobachtung* zu beschreiben. Dabei wurden Zeiten, Tätigkeiten, organisatorische Zuständigkeiten, ausführende und verantwortliche Personen sowie genutzte Informationsträger der einzelnen Aktivitäten tabellarisch aufgezeichnet.

2. Jene Abschnitte, die durch direkte Beobachtung nicht erfasst werden konnten, wurden durch *Tiefeninterviews* mit Prozessbeteiligten ergänzt.

3. Zur Validierung der Prozessbeschreibungen wurde eine *Dokumentenanalyse* der Patientenakten vorgenommen.

Weitere Details der Vorstudie hinsichtlich Methodik, Auswahl der Patienten, Zusammenfassung der Interviews etc. sind in [Sa05] dokumentiert.

Nach der Datenerhebung erfolgte im zweiten Schritt die Herleitung von Anforderungen an eine Methode zur Modellierung von Behandlungsprozessen. Dabei wurden die erhobenen Prozesse auf wiederkehrende Sachverhalte untersucht, die die Grundlage zur Definition der Anforderungen bildeten. Der dabei zusammengestellte Anforderungskatalog wird im nächsten Abschnitt vorgestellt.

4 Evaluation der EPK

4.1 Anforderungen der Gesundheitsdomäne

Nachfolgend werden 11 Anforderungen der Krankenhausdomäne an eine Modellierungsmethode dargestellt. Anforderungen 1-5 beziehen sich auf den Kontrollfluss, Anforderungen 6-8 auf das Prozessobjekt und Anforderungen 9 und 10 auf die Organisationssicht. Diese Anforderungen dürfen allerdings nicht dahingehend interpretiert werden, dass sie nur in dieser Domäne anzutreffen sind. Vielmehr wurden alle aus den erhobenen Behandlungsprozessen identifizierten Anforderungen zusammengestellt. Sie geben damit einen Gesamtüberblick über zu beschreibende Sachverhalte, anhand dessen die Ausdruckfähigkeit der EPK überprüft werden soll.

Anforderung 1: Abbildung sequentieller Prozessabläufe

Innerhalb von Behandlungsprozessen existieren streng-sequentielle Aktivitätsfolgen (Anforderung 1a), deren einzelne Aktivitäten nacheinander ausgeführt und jeweils durch das Beenden der vorangehenden Aktivität angestoßen werden. Daneben existieren wahlfrei-sequentielle Aktivitätsfolgen (Anforderung 1b). Damit sind Aktivitätsfolgen gemeint, die zwar alle innerhalb einer bestimmten Zeitspanne durchzuführen sind, jedoch aufgrund der Irrelevanz der Ausführungsabfolge wahlfrei nacheinander durchgeführt werden können.

Anforderung 2: Abbildung paralleler Prozessabläufe

Bestimmte Aktivitäten erfordern ein paralleles Ausführen. Dabei lässt sich zwischen Parallelitäten unterscheiden, bei denen keine Synchronisation erfolgt, d.h. in denen die parallelen Aktivitäten nicht wieder zusammengeführt werden (Anforderung 2a), und Parallelitäten, in denen die getrennten Aktivitätspfade nach ihrer jeweiligen Beendigung wieder synchronisiert werden müssen, d.h. in denen der Prozessablauf erst nach Fertigstellung aller Aktivitätsfolgen weitergeführt wird (Anforderung 2b).

Anforderung 3: Abbildung von Entscheidungen

Die Möglichkeit der inhaltlich adäquaten Abbildung von Entscheidungssituationen ist eine wesentliche Anforderung an die Prozessmodellierung – nicht nur in der Krankenhausdomäne. Die beobachteten Entscheidungsarten lassen sich grundsätzlich in regelbasiert (Anforderung 3a) und nicht-regelbasiert (Anforderung 3b) differenzieren. Regelbasierte Entscheidungen werden auf der Basis deterministischer und scharfer Gesetzmäßigkeiten getroffen, während nicht-regelbasierte Entscheidungen nicht-deterministisch und/oder unscharf sind.

Weiterhin können Entscheidungen hinsichtlich der zu verfolgenden Alternativen unterschieden werden. So kann es sich um eine streng-alternative Entscheidung (genau einer der Pfade wird weiterverfolgt), eine halboffen-alternative Entscheidung (Anforderung einer oder mehrere Pfade müssen zwingend, weitere können verfolgt werden) oder eine

offen-alternative Entscheidung (einer, mehrere oder alle Pfade können verfolgt werden) handeln.

Anforderung 4 : Abbildung iterativer Prozessabläufe

Untersuchungs-, Behandlungs- und Versorgungsschleifen bilden ein weiteres zentrales Phänomen von Behandlungsprozessen, da während des stationären Aufenthaltes eines Patienten bestimmte Aktivitäten in bestimmten Zyklen, Abständen oder bis zur Erreichung eines bestimmten Erfüllungsgrades iterativ zu wiederholen sind. Anforderung 4 besteht demnach in der Forderung nach

1. einer dedizierten Möglichkeit zur Modellierung iterativer Prozessabläufe,
2. einer Möglichkeit zur Definition von Zeiträumen oder Zeitpunkten der Iteration sowie
3. einer Möglichkeit zur Definition von Ein- und Austrittsbedingungen von Iterationen.

Anforderung 5: Abbildung interner und externer Prozessschnittstellen

Behandlungsprozesse sind oft stellen- und organisationsübergreifend definiert. Dabei kann unterschieden werden zwischen

1. reinen Transportschnittstellen, die den temporären Übergang eines Patienten (bspw. für eine Untersuchung) zwischen zwei Organisationseinheiten beschreiben, sowie
2. Verlegungsschnittstellen, die den vollständigen Übergang des Patienten zwischen zwei Organisationseinheiten beschreiben. Diese können weiter untergliedert werden in
   a. interne Verlegungsschnittstellen, wo der Übergang zwischen zwei Einheiten innerhalb eines Krankenhauses erfolgt und
   b. externe Verlegungsschnittstellen, wo der Übergang zu einer externen Einheit erfolgt (z. B. Verlegung in ein anderes Krankenhaus).

Bei beiden Arten ist wiederum zu unterscheiden zwischen den

1. durch den Patienten selbst ausgeführte Eigentransporte und
2. durch einen Transportdienst ausgeführte, nichtselbständige Transporte (bspw. nach einem operativen Eingriff oder bei Immobilität des Patienten).

Anforderung 6: Warte- und Ruhezustände

Bestimmte Teile der Prozesse bestehen aus Warte- und Ruhezeiten, in denen Patienten auf eine Untersuchung warten, beobachtet werden oder ruhen. Anforderung 6 besteht in der Forderung nach einer adäquaten Wiedergabe dieser passiven Zustände eines Patienten.

Anforderung 7: Extern-verantwortete Änderungen des Patientenzustands

Aktivitäten innerhalb von Behandlungsprozessen sind i.d.R. extern-verantwortete und -verursachte Einwirkungen auf den Zustand des Patienten. Dies wird beispielsweise

durch Pflegemaßnahmen oder medikamentöse bzw. operative Eingriffe erreicht. Das Erreichen eines neuen Zustands kann wiederum als Bedingung, und die Zustandsänderung selbst als auslösendes Ereignis für Aktivitäten, Entscheidungen oder Iterationen dienen.

Anforderung 8: Interne Dynamik des Prozessobjekts "Patient"

Während auch bei der Bearbeitung industrieller Güter Ruhe- und Wartezeiten sowie externe Beeinflussungen existieren, also einseitige Auswirkungen der Prozessaktivitäten auf den Prozessobjektzustand, so kann das Prozessobjekt eines Behandlungsprozesses – der Patient – auch interne Zustandsänderungen erfahren. Der Gesundheitszustand des Patienten kann sich spontan ändern, was gravierende Auswirkungen auf den weiteren Prozessablauf haben kann, weshalb darauf adäquat reagiert werden muss.

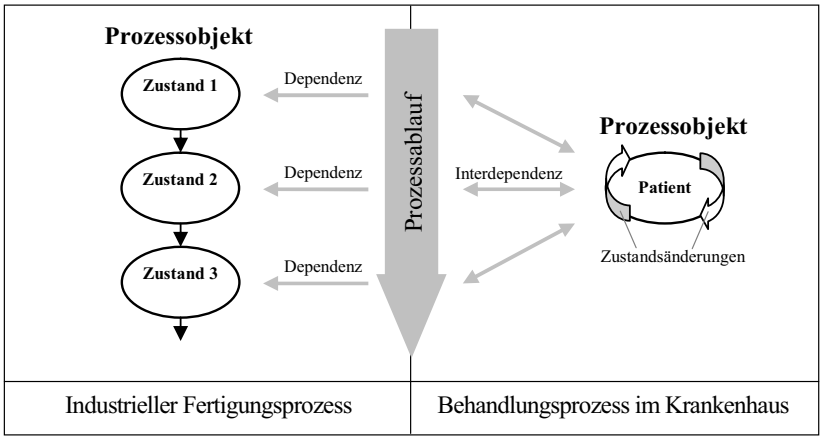

Abbildung 1: Prozessobjekt industrieller Prozessen und Behandlungsprozesse

Eine Modellierungsmethode muss demnach in der Lage sein, diese internen Änderungen des Gesundheitszustands des Patienten abzubilden, indem

1. das Prozessobjekt (als Repräsentation des Patienten) selbst spontane oder kontinuierliche Zustandsänderungen erfahren kann und
2. diese Zustandsänderungen Auswirkungen auf den Prozessablauf haben können.

Anforderung 9: Abbildung der Organisationsstruktur

Die Krankenhausrealität ist geprägt von scharfen organisatorischen Trennungen. So führt beispielsweise die alleinige Entscheidungs- und Weisungsbefugnis der Ärzte bei medizinischen Entscheidungen dazu, dass die Zuweisung von organisatorischen Zuständigkeiten zu Aktivitäten eine wichtige Rolle spielt. Demnach sollte ein Sprachkonzept zur Modellierung von Behandlungsprozessen in der Lage sein

1. den Verantwortungsbereich (Pflege / Medizin / Verwaltung) abzubilden,
2. den einzelnen Aktivitäten und Entscheidungen die jeweils ausführende, anweisende und verantwortliche organisatorische Einheit zuzuordnen und

3. weisungstechnische Abhängigkeiten, Zielkonflikte und andere aufbauorganisatorische Gegebenheiten der am Prozess direkt oder indirekt beteiligten Akteure wiederzugeben.

Anforderung 10: Abbildung technischer Ressourcen

Viele Aktivitäten innerhalb eines Behandlungsprozesses benötigen neben den ausführenden personellen Ressourcen zusätzlich bestimmte technische Ressourcen. Dies können zum einen Gerätschaften oder Verbrauchsmaterial sein, zum anderen können sie sich auf die Belegung räumlicher Kapazitäten beziehen. Um Kapazitätskonflikte parallel laufender Prozesse zu verhindern und eine effiziente Ressourcenzuordnung zu ermöglichen, sollte eine Modellierungssprache diese abbilden können.

Anforderung 11: Abbildung von Zeiten

Die Forderung nach einer Integration der zeitlichen Perspektive ist in deren hoher Bedeutung für viele Aspekte der Ausführung von Behandlungsprozessen begründet. Die hauptsächlichen Aktivitäten des Behandlungsprozesses spielen sich, von Notfällen einmal abgesehen, innerhalb bestimmter Tageszeiten (früher Morgen bis später Nachmittag) ab. Dies bedeutet beispielsweise, dass je nachdem zu welcher Uhrzeit ein Behandlungsprozess beginnt, bestimmte Aktivitäten noch am selben Tag stattfinden können, oder aber erst am nachfolgenden Tag, was zu einer Verlängerung des gesamten Klinikaufenthalts führen kann.

Darüber hinaus initiiert eine solche zeitliche Verschiebung weitere Aktivitäten, wie zusätzliche Pflegemaßnahmen (Verpflegung, Überwachung, etc.) und beeinflusst somit den gesamten Prozessablauf. Auch bei iterativen Prozessabläufen (als Eintrittsbedingung) und bei Entscheidungen (als Kriterium) spielen Zeiten eine wichtige Rolle. Die übergeordnete Forderung nach einer Wiedergabe von Zeiten lässt sich in folgende Teilanforderungen unterteilen:

1. Der Einfluss von Zeiten auf die Behandlungsaktivitäten muss abbildbar sein.
2. Zeiten müssen als Eintrittsbedingung und Entscheidungskriterium definierbar sein.
3. Aktivitäten müssen Zeiten bzw. Zeitspannen zugeordnet werden können.

Nachfolgende Tabelle 3 fasst die 11 Anforderungen an die Modellierung von Behandlungsprozessen in einem Anforderungskatalog zusammen, der die Ausgangsbasis für die Evaluation der EPK bildet.

Sichten	Nr.	Anforderung		
Kontroll-fluss	1	Abbildung sequentieller Prozessabläufe	1a) Streng sequentiell	
			1b) Wahlfrei sequentiell (lokale Autonomie)	
	2	Abbildung paralleler Prozessabläufe	2a) mit synchronisierter Zusammenführung	
			2b) ohne synchronisierte Zusammenführung	
	3	Abbildung von Entscheidungen	3a) Regelbasiert (scharf)	$3a_1$) Offen-alternativ
				$3a_2$) Halboffen-alternativ
				$3a_3$) Streng-alternativ
			3b) Nicht regelbasiert (unscharf)	$3b_1$) Offen-alternativ
				$3b_2$) Halboffen-alternativ
				$3b_3$) Streng-alternativ
	4	Abbildung iterativer Prozessabläufe		
	5	Abbildung der internen und externen Prozessschnittstellen		
Prozessobjekt (Patient)	6	Abbildung von Warte- und Ruhezuständen		
	7	Abbildung externer Einwirkungen auf den Patientenzustand		
	8	Abbildung der internen Dynamik des Prozessobjekts		
Ressourcen	9	Abbildung der Organisationsstruktur		
	10	Abbildung technischer Ressourcen		
	11	Abbildung von Zeiten		

Tabelle 3: Anforderungen an die Modellierung von Behandlungsprozessen

4.2 EPK und EPK-Erweiterungen

Mit zunehmender Diskussion und betrieblichem Einsatz wurde die Konstrukte der EPK (Funktionen, Ereignisse sowie die Konnektoren AND, XOR und OR) um eine Vielzahl unterschiedlicher Konstrukte und Varianten erweitert. Eine vollständige Aufarbeitung ist in der aktuellen Literatur nicht auszumachen. Somit erfolgt in Tabelle 4 eine möglichst umfassende Auflistung und Klassifizierung der Erweiterungen und Varianten, jedoch ohne Anspruch auf Vollständigkeit. Diese werden im nächsten Kapitel gemeinsam mit der ursprünglich definierten EPK der Methodenevaluation unterzogen.

Nr.	Erw.	Name / Inhalt der Erweiterung	Autor / Quelle	Jahr
1	EPK-Varianten	Real-Time Variante (rEPK)	HOFFMANN et al. [Ho93]	1993
2		Objektorientierte Variante (oEPK)	SCHEER et al. [Sc97]	1997
3		Modified EPCs (modEPK)	RITTGEN [Ri99]	1999
4		Agentenorientierte Variante (xEPK)	KIRN et al. [Ki00]	2000
5	Erweiterung um Konnektoren	SEQ-Konnektor	PRIEMER [Pr95]	1995
6		ET-Konnektor	ROSEMANN [Ro96]	1996
7		OR_1-Konnektor		
8		XORUND-Konnektor	RITTGEN [Ri99]	1999
9		Fuzzy-Konnektor (Fuzzy-EPK)	THOMAS et al. [THA02]	2002
10		Multi-Instanziierungs-Konnektoren	RODENHAGEN [Ro02]	2002
11		Empty-Konnektor (yEPK)	MENDLING et al. [MNN05]	2005
12	Erweiterung um sonstige Konstrukte	Erweiterte EPK (eEPK)	-	-
13		Prozessobjekte	ROSEMANN [Ro96]	1996
14		Ereignis-Zustands-Konstrukte	SCHÜTTE [Sc98]	1998
15		Visualisierung der Organisationssicht	SCHÜPPLER [Sc98]	1998
16		Erweiterung um UML-Elemente	LOOS; ALLWEYER [LA98]	1998
17		Nachrichtensteuerung	SCHEER [Sc02]	2002
18		Prozessobjektmigrationfunktion	KUGELER [Ku02]	2002
19		Erweiterung zum Risikomanagement	BRABÄNDER; OCHS [BO02]	2002
20		Visualisierung der Kundenintegration	SCHNEIDER; THOMAS [ST03]	2003
21		Interorganisationale Erweiterungen	KLEIN et al. [KKS04]	2004
22		Multiple Instantiation (yEPK)	MENDLING et al. [MNN05]	2005
23		Cancellation (yEPK)		

Tabelle 4: Klassifizierung der EPK-Erweiterungen und –Varianten

4.3 Einsatzpotentiale der EPK

Um die EPK bezüglich der Modellierung von Behandlungsprozessen zu untersuchen, wurde geprüft, inwieweit die EPK und vorgeschlagene EPK-Erweiterungen in der Lage sind, die geforderten Anforderungen zu beschreiben. Während bei einigen EPK-Konstrukten die Anforderungserfüllung von vornherein ausgeschlossen werden konnte, war bei vielen Kombinationen eine genauere Anforderungsprüfung notwendig. Nachfolgend wird die Anforderungsprüfung aus Platzgründen anhand von zwei Beispielen demonstriert.

Beispiel 1: Überprüfung der Anforderung 1b: wahlfrei-sequentielle Prozessabläufe

Die in der lokalen Handlungsautonomie begründete, wahlfreie Durchführung bestimmter Aktivitätssequenzen kann bereits durch den XOR-Konnektor der Basis-EPK realisiert werden. Durch die Kombinatorik der Möglichkeiten kann diese Form der Modellierung bei längeren wahlfreien Sequenzen jedoch relativ komplex und unübersichtlich werden. Alternativ kann die Anforderung mit Hilfe des von PRIEMER [Pr95] und ROSEMANN [Ro95] vorgeschlagenen Sequenz-Konnektors (SEQ-Konnektor) realisiert werden. Um diesen erweitert ist die EPK in der Lage, wahlfreie Funktionsabfolgen ohne ablaufindividuelle Eigenschaften innerhalb einer Prozesskette abzubilden, und somit die lokale Handlungsautonomie adäquat zu unterstützen (vgl. Abbildung 1).

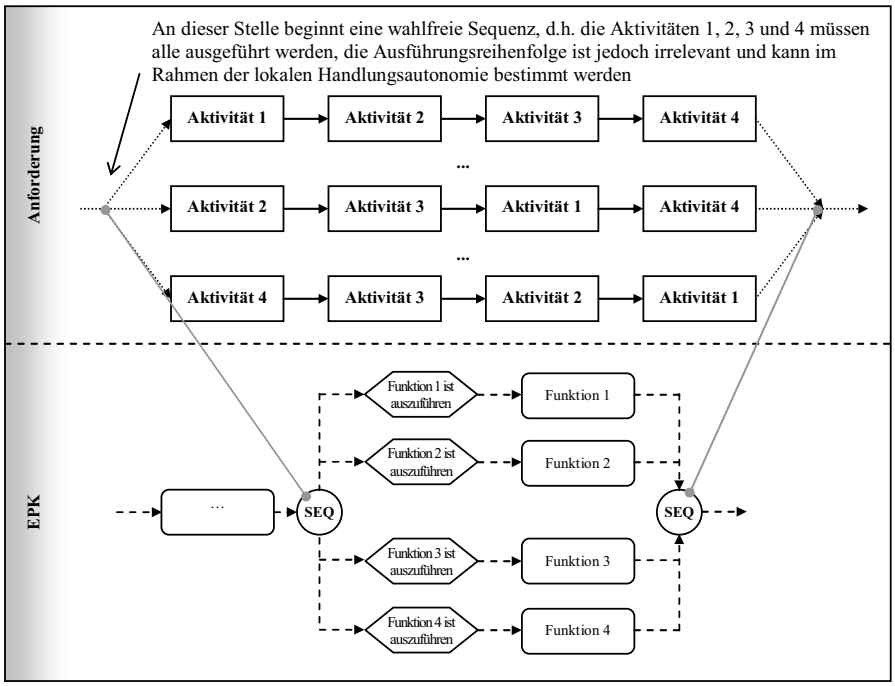

Abbildung 1: Überprüfung der Anforderung 1b: Abbildung wahlfrei-sequentieller Abläufe

Beispiel 2: Überprüfung der Anforderung 4: Abbildung iterativer Prozessabläufe

Untersuchungs-, Behandlungs- und Versorgungsschleifen bilden ein zentrales Phänomen von Behandlungsprozessen. Deren explizite Darstellung – d.h. die Abbildung einer zyklisch auszuführenden Aktivität ohne deren redundante Modellierung an bestimmten Punkten in einen Prozessablauf – wird von der ursprünglichen EPK jedoch nicht unterstützt. Im Rahmen der agentenorientierten EPK-Variante *xEPK* (welche aus ihrer Ent-

stehung heraus einen engen Bezug zu Krankenhausprozessen besitzt) schlagen KIRN et al. [Ki00] ein Konstrukt zur expliziten Modellierung iterativer Prozesse vor.

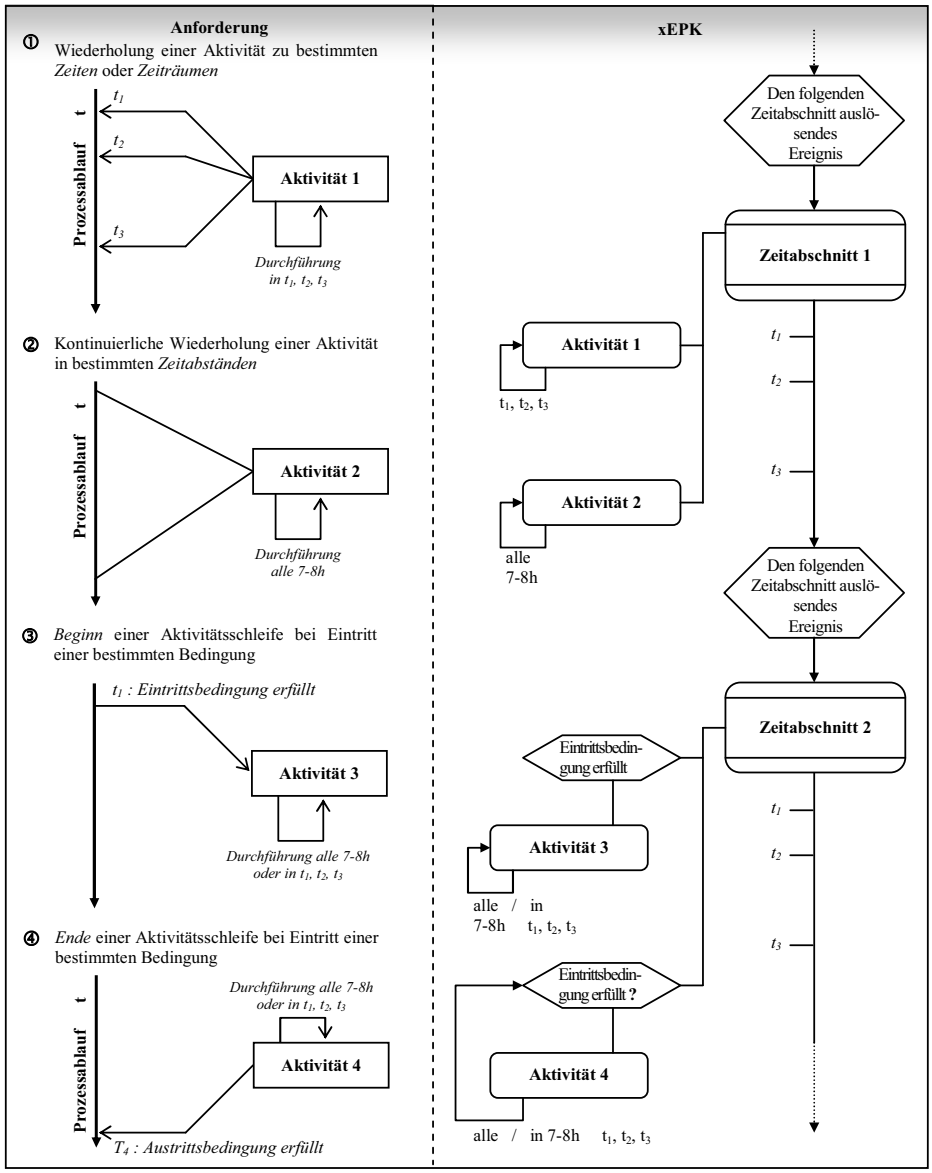

Abbildung 2: Überprüfung der Anforderung 4: Abbildung iterativer Prozessabläufe

Diese Schleifendarstellung kann aber nicht ohne weiteres in EPK-Modelle integriert werden. Sie werden (wie alle anderen Funktionen der xEPK auch) als Methode an ein

bestimmtes Zeitfenster – eine aggregierte Funktion, bspw. einen Behandlungstag – gekoppelt. Sie bieten jedoch die Möglichkeit, die geforderten Ein- und Austrittsbedingungen sowie verschiedene zeitliche Ausführungsrelationen (Ausführung zu bestimmten Uhrzeiten, Ausführung in bestimmten Zeitabständen) zu modellieren (vgl. Abbildung 2).

Dadurch, dass die Schleifenkonstrukte stets an die Funktion gekoppelt werden, wird von einer zeitlichen Anordnung oder Reihenfolgenbildung der Aktivitäten abgesehen. Die geforderte lokale Handlungsautonomie bezüglich der Integration der Schleifenaktivitäten wird somit ebenfalls implizit unterstützt. Insofern kann die Anforderung 4 prinzipiell durch die xEPK erfüllt werden, wobei die besondere Syntax und Semantik dieser Variante berücksichtigt werden muss.

5 Gesamtbeurteilung der EPK

Die Ergebnisse der Prüfung der einzelnen Anforderungen ist im Anhang tabellarisch zusammengefasst. Die Anforderungen bezüglich des Kontrollflusses und der Abbildung von Ressourcen konnten zu einem großen Teil durch Konstrukte der EPK erfüllt werden. Hingegen bestehen bezüglich adäquater Abbildung des Prozessobjekts, der Modellierung nicht-regelbasierter Entscheidungen sowie der Abbildung von Zeiten und unterschiedlicher Iterationsvarianten noch Defizite.

Die Kontrollflussperspektive ist der einzige Bereich, in der bereits die ursprüngliche EPK in der Lage ist, bestimmte Anforderungen zu erfüllen. So können die Anforderungen 1 und 2, die Abbildung sequentieller und paralleler Prozessabläufe, auch bezüglich aller Teilanforderungen, als vollständig erfüllt angesehen werden. Und auch der Bereich 3a, die Abbildung verschiedener regelbasierter Entscheidungen, kann durch Konstrukte der ursprünglichen EPK abgedeckt werden.

Das erste Erweiterungspotential existiert im Bereich der nicht-regelbasierten Entscheidungen (Anforderung 3b). Zwar existieren diese auch in Unternehmen, weshalb die EPK um Konstrukte wie den Fuzzy-Konnektor erweitert wurde ([THA02], [Hü03]) jedoch unterscheiden sie sich von jenen innerhalb von Behandlungsprozessen. Diese werden auch von Autoren thematisiert, die die Anwendung der Fuzzy-Set-Theorie zur Beschreibung von Entscheidungen im medizinischen Umfeld thematisierten (vgl. z.B. [Hu97], [St97], [Re04]). Während die Unschärfe in Unternehmen bei zunehmender Kenntnis der die Entscheidungssituation bestimmenden Variablen annährend lösbar ist, fehlt diese Kenntnis bei klinischen Entscheidungen oftmals. Einige Entscheidungen werden von Ärzten getroffen, ohne dass unmittelbar klar ist, welche Variablen die Entscheidung maßgeblich beeinflusst haben. Diese nicht-deterministischen (intuitiven) Entscheidungen können z.B. von persönlichen Einschätzungen eines Arztes abhängen, ohne dass die Entscheidungsgrundlage selbst vom Arzt expliziert werden kann. Diese Art von Entscheidungen beschränkt sich nicht nur auf die Arbeit von Ärzten. Auch bei pflegerischen Aktivitäten konnten, in wesentlich engeren Grenzen als bei Ärzten, unscharfe und intuitive Entscheidungssituation beobachtet und dokumentiert werden. Ganz anders sieht die Situation bei Verwaltungstätigkeiten aus, wo für getroffene Entscheidungen meist entsprechende Vorgaben vorliegen.

Behandlungsschleifen und die daraus resultierende Anforderung nach einer dedizierten Abbildung iterativer Prozessabläufe konnte lediglich durch die agentenorientierte EPK – die xEPK – weitgehend gelöst werden. Die stark von der EPK abweichende Semantik des xEPK-Ansatzes macht allerdings noch weitere Untersuchungen notwendig.

Prozessschnittstellen lassen sich unter anderem durch die Schnittstellendiagramme der interorganisationalen Erweiterungen nach KLEIN, KUPSCH und SCHEER [KKS04] beschreiben, und hinsichtlich verschiedener Schnittstellenarten differenzieren, so dass auch dieser Bereich zufriedenstellend abgedeckt werden kann.

Die zweite Anforderungsperspektive, die Abbildung des Prozessobjekts, macht deutlich, dass sich Patienten in ihrem Zustandsverhalten von den Prozessobjekten in Unternehmen unterscheiden. Bereits bei der Abbildung von Warte- und Ruhezuständen sowie externer Zustandsbeeinflussungen des Patienten, welche aufgrund ähnlicher Vorgänge in Geschäftsprozessen prinzipiell modellierbar sein sollten, stoßen relevante EPK-Erweiterungen (rEPK und die UML-Erweiterung um State-Chart-Diagramme) an ihre Grenzen. Gleiches gilt in höherem Maße für die Anforderung 8, welche die Abbildung der internen Dynamik des Patientenzustands verlangt. Dass sich der Prozessobjektzustand spontan und initiativ auf eine Art und Weise ändern kann, welche den gesamten weiteren Prozessablauf beeinflussen kann, ist ein spezielles Phänomen von Behandlungsprozessen und somit ein Novum für die EPK. So ist bislang noch kein EPK-Konstrukt in der Lage, die entstehende Interdependenz zwischen Prozessablauf und Prozessobjekt adäquat wiederzugeben.

Die dritte Perspektive, die Abbildung von Ressourcen, lässt sich wiederum mit der EPK vergleichsweise gut abbilden, da vor allem bezüglich einer Beschreibung der Organisationssicht eine ganze Reihe an Erweiterungen vorgeschlagen wurde.

Der letzte Bereich, die Abbildung der zeitlichen Perspektive, kann durch die EPK nur teilweise erfüllt werden. Zwar könnten Uhrzeiten indirekt als Bedingungen zur Entscheidungsfindung dienen, ohne eine explizite Zeitmodellierung vorzunehmen. Auch die Definition von Zeitdauern bestimmter Funktionen oder Prozessabschnitte stellt keine unüberwindbare Herausforderung dar. Doch das spezifische zeitliche Phänomen von Behandlungsprozessen, die uhrzeitliche Abhängigkeit des Prozessablaufs, lässt sich nur durch redundante Modellierung aller Prozessabschnitte in Abhängigkeit der jeweiligen Uhrzeit lösen, was jedoch aufgrund der Kombinationsvielfalt zu komplexen und wenig handhabbaren Modellen führen kann. Insofern fehlt der EPK hier ein dediziertes Konstrukt, welches den Einfluss der Uhrzeit auf den Prozessablauf abbilden kann.

6 Zusammenfassung und Ausblick

Primäres Ziel der Arbeit war die Überprüfung der Anwendbarkeit der EPK zur Modellierung von Behandlungsprozessen. Zur Beantwortung dieser Frage wurde, durch die Analyse empirisch erhobener Daten, ein induktiv-hergeleiteter Anforderungskatalog erstellt. Dieser Anforderungskatalog ist das erste Ergebnis der vorliegenden Arbeit, und kann, unabhängig von der im Weiteren diskutierten EPK-Evaluation, als eigenständige Grund-

lage für weitere Forschungsbemühungen der Prozessmodellierung im Krankhausumfeld dienen. Basierend auf diesem Anforderungskatalog wurde die EPK evaluiert, um die Frage nach ihrer Eignung zur Abbildung von Behandlungsprozessen zu klären.

Bei der Untersuchung wurde deutlich, dass die EPK und die vorgeschlagenen Erweiterungen den regelbasierten Kontrollfluss und die beteiligten Ressourcen relativ gut abbilden können. Die EPK eignet sich, wie Eingangs vermutet, für die Beschreibung von strukturierten Verwaltungstätigkeiten. Um die Ausdrucksmächtigkeit der EPK an Anforderungen der Krankenhausdomäne anzupassen, existiert jedoch noch Erweiterungspotential in den folgenden drei Bereichen, um neben Verwaltungstätigkeiten auch pflegerische und ärztliche Tätigkeiten als integrale Bestandteile eine Behandlungsprozesses adäquat abbilden zu können:

1. die Wiedergabe nicht-regelbasierter und z.T. intuitiver Entscheidungen bei ärztlichen und z.T. pflegerische Aktivitäten,
2. die Modellierung der internen Zustandsdynamik des Patienten und dessen Einfluss auf den Prozessablauf,
3. die dimensionsübergreifende Abbildung der zeitlichen Perspektive von Aktivitäten, sowie
4. Abbildung unterschiedlicher Iterationsvarianten.

Diese Defizite stellen Anknüpfungspunkte für weiteren Forschungsbedarf hinsichtlich der Erweiterung der EPK.

Die vorgestellten Ergebnisse müssen vor dem Hintergrund der Limitation der Untersuchung interpretiert werden. Diese liegt in erster Linie in der Subjektivität von Entscheidungen, die im Rahmen der Evaluation zu treffen waren. Bereits die Erhebung der empirischen Datenbasis durch Beobachtung, auf der die Anforderungsableitung letztlich basiert, ist durch die Subjektivität der Methoden der Feldforschung geprägt. Diese konnte allerdings durch den Informationsaustausch zwischen Studienmitarbeitern und Klinikpersonal durch Tiefeninterviews, sowie durch die anschließende Validierung der erhobenen Daten durch die Patientenakten (Dokumentenanalyse) begrenzt werden. Bei der Auswertung der Daten ließen sich stellenweise auch andere Anforderungen ableiten und daraus wiederum andere Schlussfolgerungen hinsichtlich der Erweiterung der EPK zur Modellierungssprache für Behandlungsprozesse ziehen.

Anhang

			Varianten				Erw. um Konnektoren							Erw. um sonstige Konstrukte											
Anforderungen		EPK	(rEPK)	oEPK	Modified EPCs	xEPK	SEQ	ET	OR_1	XORUND	Fuzzy	Multiinstanzen	Empty	eEPK	Prozessobj.	E-Z-Konstrukte	Vis. Org-Sicht	UML Erw.	Nachrichten	Prozessob. migr.	Risikoman.	Kundenintegr.	Interorg. Erw.	Multiple Inst.	Cancellation	**Bester Wert**
Kontrollfluss	1a	X																							**X**
	1b	X					X																			**X**
	2a	X											X													**X**
	2b	X																								**X**
	$3a_1$	X						X		X																**X**
	$3a_2$	X						X	X																	**X**
	$3a_3$	X						X																		**X**
	$3b_1$	O									/															/
	$3b_2$	O									/															/
	$3b_3$	O									/															/
	4	O				/																				/
	5	O												/			/						X			**X**
Prozess-objekt	6	O	/												/			/		/							/
	7	O	/												/			/		/							/
	8	O																								**O**
Ress.	9	O												/			/	X					/			**X**
	10	O												/												/
	11	/																								/

X : erfüllt, / : teilweise erfüllt, O : nicht erfüllt, leer : nicht relevant

Tabelle 5: Ergebnisse der empirischen Methodenevaluation
(für Anforderungen vgl. Tabelle 3)

Literaturverzeichnis

[Be04] Berger, K.: Behandlungspfade als Managementinstrument im Krankenhaus, in: Greiling, M. (Hrsg.): Pfade durch das Klinische Prozessmanagement: Methodik und aktuelle Diskussionen, Stuttgart: Kohlhammer 2004, S. 42-64.

[BO02] Brabänder, E.; Ochs, H.: Analyse und Gestaltung prozessorientierter Risikomanagementsysteme mit Ereignisgesteuerten Prozessketten, in: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des 1. GI-Workshop EPK 2002, Trier 2002, S. 17-34.

[Br05] Braun, G. E.: Zunehmende Prozessorientierung als Entwicklungstendenz im gesundheitspolitischen Umfeld des Krankenhauses. in: G. E. Braun; N. Bohr; O. Braun Güssow (Hrsg.): Prozessorientiertes Krankenhaus : Lösungen für eine Positionierung im Wettbewerb. Stuttgart 2005, S. 13-28.

[DR00] Dadam, P.; Reichert, M.: Towards a New Dimension in Clinical Information Processing, in: Hasman, A. et al. (Hrsg.): Medical Infobahn for Europe, Amsterdam: IOS Press 2000, S. 295-301.

[Dy02] Dykes, P. C.: Die Bearbeitung von Begleiterkrankungen und Abweichungen mit "Co-Pathways", in: Dykes, P. C.; Wheeler, K. (Hrsg.): Critical Pathways –Interdisziplinäre Versorgungspfade – DRG-Management Instrumente, Bern et al.: Huber 2002, S. 57-66.

[Fl86] Floyd, C.: A comparative evaluation of system development methods. Proc. of the IFIP WG 8.1 working conference on Information systems design methodologies: improving the practice table of contents. Noordwijkerhout, Netherlands 1986, S. 19 - 54.

[GH02] Greiling, M.; Hofstetter, J.: Patientenbehandlungspfade optimieren – Prozeßmanagement im Krankenhaus, Kulmbach: Baumann 2002.

[Go05] Gospodaravskaya, E.; Churilov, L.; Wallace, L.: Modelling the Patient Care Process of an Acute Care Ward in a Public Hospital: A Methodological Perspective, in: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 6 - Volume 06 2005, S. 145.1.

[Gr99] Grote, W.; Bott, O. J.; Penger, O.-S.; Terstappen, A.; Pretschner1, D. P.; F.Kloß: Prozessorientierte Präsentation, Evaluation und Einführung von Anwendungssystemen am Beispiel des OP-Managements, in: Victor, N. et al. (Hrsg.): Medical Informatics, Biostatistics and Epidemiology for Efficient Health Care and Medical Research, 44. Jahrestagung der GMDS in Heidelberg, September 1999, München: Urban und Vogel 1999, S. 252-255.

[Ho93] Hoffmann, W.; Wein, R.; Scheer, A.-W.: Konzeption eines Steuerungsmodells für Informationssysteme – Basis für die Real-Time-Erweiterung der EPK (rEPK), Heft 106 des Instituts für Wirtschaftsinformatik im Institut für empirische Wirtschaftsforschung an der Universität des Saarlandes, Saarbrücken 1992.

[Hu97] Hurdle, J. F.: Lightweight fuzzy processes in clinical computing. In: Artificial Intelligence in Medicine 11 (1997) 1, S. 55-73.

[Hü03] Hüsselmann, C.: Fuzzy-Geschäftsprozessmanagement. Josef Eul Verlag, Lohmar 2003.

[Jø03] Jørgensen, J. B.: Coloured Petri Nets in Development of a Pervasive Health Care System, in: van der Aalst, W. M. P.; Best, E. (Hrsg.): Proceedings of ICATPN 2003, LNCS 2679, Springer: Berlin; Heidelberg 2003, S. 256–275.

[Ke92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)", Heft 89 des Instituts für Wirtschaftsinformatik im Institut für empirische Wirtschaftsforschung an der Universität des Saarlandes, Saarbrücken 1992.

[Ke99] Keller, G.: SAP R/3 prozeßorientiert anwenden. Iteratives Prozeß-Prototyping mit Prozeßketten. Bd. 3, München 1999.

[Ki00] Kirn, S.; Heine, C.; Petsch, M.; Puppe, F.; Klügl, F.; Herrler, R.: Agentenorientierte Modellierung vernetzter Logistikkreisläufe als Ausgangspunkt agentenbasierter Simulation, in: Kirn, S.; Petsch, M. (Hrsg.): Tagungsband zum 2. Kolloquium des DFG-Schwerpunktprogramms "Intelligente Softwareagenten und Betriebswirtschaftliche Anwendungsszenarien", TU Ilmenau, Lehrstuhl für Wirtschaftsinformatik 2, Ilmenau: November 2000.

[KKS04] Klein, R.; Kupsch, F.; Scheer, A.-W.: Modellierung interorganisationaler Prozesse mit Ereignisgesteuerten Prozessketten, Heft 178 des Instituts für Wirtschaftsinformatik im Institut für empirische Wirtschaftsforschung an der Universität des Saarlandes, Saarbrücken 2004.

[Ku02] Kugeler, M.: SCM und CRM – Prozessmodellierung für Extended Enterprises, in: Becker, J.; Kugeler, M.; Rosemann, M. (Hrsg.): Prozessmanagement – Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 3., vollständig neubearbeitete und erweiterte Auflage, Berlin et al.: Springer 2002, S. 457-485.

[Kü04] Küttner, T.: Der Klinische Behandlungspfad als strategisches Managementinstrument im DRG-Kontext und dessen Entwicklung am Beispiel einer akutgeriatrischen Abteilung eines somatischen Krankenhauses, Schüling: Münster 2004.

[LA98] Loos, P.; Allweyer, T.: Process Orientation and Objekt-Orientation – An Approach for Integrating UML and Event-Driven Process Chains (EPC), Heft 144 des Instituts für Wirtschaftsinformatik im Institut für empirische Wirtschaftsforschung an der Universität des Saarlandes, Saarbrücken 1998.

[Lo92] Loos, P.: Datenstrukturierung in der Fertigung. Oldenbourg Verlag; München [u.a.] 1992.

[Lü03] Lüngen, M.; Lauterbach, K. W.; Lüngen, L.: DRG in deutschen Krankenhäusern : Umsetzung und Auswirkungen. Stuttgart [u.a.] 2003.

[Mi01] Middendorf, C.: Ganzheitliches Medical Controlling, in: von Eiff, W.; Ziegenbein, R. (Hrsg.): Geschäftsprozeßmanagement – Methoden und Techniken für das Management von Leistungsprozessen im Krankenhaus, 2. Auflage, Gütersloh: Bertelsmann Stiftung 2003, S. 161-219.

[MNN05] J. Mendling, G. Neumann, M. Nüttgens: Yet Another Event-driven Process Chain - Modeling Workflow Patterns with yEPCs. accepted for the International Journal "Enterprise Modelling and Information Systems Architectures".

[Mo01] Moser, G.: Standardprozeßorientierter Krankenhausvergleich, in: von Eiff, W.; Ziegenbein, R. (Hrsg.): Geschäftsprozeßmanagement – Methoden und Techniken für das Management von Leistungsprozessen im Krankenhaus, 2. Auflage, Gütersloh: Bertelsmann Stiftung 2003, S. 83-134.

[Pe62] Petri, C. A.: Kommunikation mit Automaten, in: Peschl, E.; Unger, H. (Hrsg.): Schriften des Rheinisch-Westfälischen Institutes für Instrumentelle Mathematik an der Universität Bonn Nr. 2, Bonn: 1962.

[Pe03] Perrevort, F.: Modellierung eines integrierten Informations- und Kommunikationssystems im Krankenhaus – ARIS in der Anwendung, in: Kuntz, L. (Hrsg.): Arbeitsberichte zum Management im Gesundheitswesen (Nr. 4), Lehrstuhl für Allgemeine BWL und Management im Gesundheitswesen, Universität zu Köln: 2003.

[Pr95] Priemer, J.: Entscheidungen über die Einsetzbarkeit von Software anhand formaler Modelle, Sinzheim: Pro Universitate 1995.

[Re97] Reichert, M.; Schultheiß, B.; Dadam, P.: Erfahrungen bei der Entwicklung vorgangsorientierter, klinischer Anwendungssysteme auf Basis prozeßorientierter Workflow-Technologie, Proceedings der 42. Jahrestagung GMDS, Ulm: 1997, S. 181-187.

[Re04] Seising, R.: Fuzzy Sets in Medicine - Historical Remarks. In: (F. Valafar; H. Valafar Hrsg.): Proceedings of the International Conference on Mathematics and Engineering

Techniques in Medicine and Biological Scienes, METMBS '04,. Las Vegas, Nevada, USA 2004, S. 31-37.

[Ri99] Rittgen, P.: Modified EPCs and Their Formal Semantics, Arbeitsbericht Nr. 19, Institut für Wirtschaftsinformatik, Koblenz: Universität Koblenz-Landau 1999.

[Ro96] Rosemann, M.: Komplexitätsmanagement in Prozessmodellen: methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung, Wiesbaden: Gabler 1996.

[Ro02] Rodenhagen, J.: Ereignisgesteuerte Prozessketten (EPK) – Multi-Instanziierungsfähigkeit und referentielle Persistenz, in: Nüttgens, M.; Rump, F. J. (Hrsg.): EPK 2002 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des GI-Workshops und Arbeitskreistreffens (Trier, November 2002), S. 95-107.

[Ro03] Roeder N.; Hindle, D.; Loskamp, N.; Juhra, C.; Hensen, P.; Bunzemeier, H.; Rochell, B.:Frischer Wind mit klinischen Behandlungspfaden (I) – Instrumente zur Verbesserung der Organisation klinischer Prozesse, in: das Krankenhaus (2003) 1, S. 20-27.

[Ru99] Rump, F. J.: Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten – Formalisierung, Analyse und Ausführung, Stuttgart; Leipzig: Teubner 1999.

[Sa05] Sarshar, K.; Loos, P.; Dominitzki, P., Reichel, C.: Krankenhausprozesse – Dokumentation erhobener Daten einer Feldstudie in einem Universitäts-Klinikum, Working Papers of the Research Group Information Systems & Management (ISSN 1617-6324 (printed version), ISSN 1617-6332 (Internet version), URN urn:nbn:de:0006-0242), Paper 24, Mainz 2005.

[Sc94] Scheer, A.-W.: ARIS-Toolset: Die Geburt eines Softwareproduktes. In: A.-W. Scheer (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik. Bd. 111, Saarbrücken 1994.

[Sc95] Scheer, A.-W.: Wirtschaftsinformatik. Referenzmodelle für industrielle Geschäftsprozesse, 6. Auflage, Berlin et al.: Springer 1995.

[Sc96] Scheer, A.-W.; Chen, R.; Zimmermann, V.: Geschäftsprozesse und integrierte Informationssysteme im Krankenhaus, Heft 130 des Instituts für Wirtschaftsinformatik im Institut für empirische Wirtschaftsforschung an der Universität des Saarlandes, Saarbrücken 1996.

[Sc97] Scheer, A.-W.; Nüttgens, M.; Zimmermann, V.: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) – Methode und Anwendung, Heft 141 des Instituts für Wirtschaftsinformatik im Institut für empirische Wirtschaftsforschung an der Universität des Saarlandes, Saarbrücken 1997.

[Sc98] Schüppler, D.: Informationsmodelle für überbetriebliche Prozesse : ein Ansatz zur Gestaltung von Interorganisationssystemen, Frankfurt am Main et al.: Lang 1998.

[Sc98] Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung, Wiesbaden: Gabler 1998.

[Sc02] Scheer, A.-W.: ARIS: Von der Vision zur praktischen Geschäftsprozesssteuerung, in: Scheer, A.-W.; Jost, W. (Hrsg.): ARIS in der Praxis – Gestaltung, Implementierung und Optimierung von Geschäftsprozessen, Berlin et al.: Springer 2002, S. 1-14.

[SL04] Sarshar, K.; Loos, P.: Klassifikation von Sprachen zur Modellierung medizinischer Behandlungspfade. In: (M. Rebstock Hrsg.): Modellierung betrieblicher Informationssyteme - MobIS 2004, Proceedings zur Tagung, 10. März 2004. Essen 2004, S. 43-59.

[SL05a] Sarshar, K.; Loos, P.: Modellierung überbetrieblicher Behandlungsprozesse durch Objekt-Petrinetze, in: Wirtschaftsinformatik 47 (2005) 3, S. 203-210.

[SL05b] Sarshar, K.; Loos, P.: A Normative Language Approach to the Application of Petri Nets in Healthcare, in: Workshop on eHealth. Workshop im Rahmen der Informatik 2005, 35. Jahrestagung der Gesellschaft für Informatik e. V..

[SR98] Siau, K.; Rossi, M.: Evaluation of Information Modeling Methods - A Review. Thirty-First Annual Hawaii International Conference on System Sciences. Volume 5: Modeling Technologies and Intelligent Systems. Kohala Coast, Hawaii, USA 1998, S. 314-322.

[St97] Steimann, F.: Fuzzy Set Theory in Medicine. In: Artificial Intelligence in Medicine 11 (1997) 1, S. 1-7.

[ST03] Schneider, K.; Thomas, O.: Kundenorientierte Dienstleistungsmodellierung mit Ereignisgesteuerten Prozessketten, in: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des GI-Workshops und Arbeitskreistreffens (Bamberg, Oktober 2003), S. 87-93.

[Su88] Surya, B. Y.; Ralph, R. B.; Akemi, T. C.; Rajkumar, T. M.: Comparison of analysis techniques for information requirement determination. In: Communications of the ACM 31 (1988) 9, S. 1090 - 1097.

[SW04] Sloane, E. B.; Wagner, W. P.: Clinical Engineering Use of Event-Driven Process Chain (EPC) to Develop Enterprise Models of Collaborative Information and Workflow for Re-Engineering the Operating Room, in: Proceedings of the 10th Americas Conference on Information Systems, New York: 2004.

[THA02] Thomas, O.; Hüsselmann, C.; Adam O.: Fuzzy-Ereignisgesteuerte Prozessketten - Geschäftsprozessmodellierung unter Berücksichtigung unscharfer Daten, in: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2002 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des GI-Workshops und Arbeitskreistreffens (Trier, November 2002), S. 7-16.

[vAa98] van der Aalst, W.M.P.: The application of Petri nets to workflow management, in: The Journal of Circuits, Systems and Computers 8 (1998) 1, S. 21-66.

[Va04] Valk, R.: Object Petri Nets - Using the Nets-within-Nets Paradigm. In: J. Desel; W. Reisig; G. Rozenberg (Hrsg.): Lectures on Concurrency and Petri Nets: Advances in Petri Nets. 3098. Aufl., 2004, S. 819-848.

[vEZ03] von Eiff, W.; Ziegenbein, R.: Entwicklung von Prozeßmodellen im Krankenhaus, in: von Eiff, W.; Ziegenbein, R. (Hrsg.): Geschäftsprozeßmanagement – Methoden und Techniken für das Management von Leistungsprozessen im Krankenhaus, 2. Auflage, Gütersloh: Bertelsmann Stiftung 2003, S. 55-83.

[Wa02] Wang, D.; Peleg, M.; Tu, S. W.; Boxwala, A. A.; Greenes, R. A.; Patel, V. L.; Shortliffe, E. H.: Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines: A literature review of guideline representation models. In: International Journal of Medical Informatics 68 (2002) 1-3, S. 59-70.

Meta-Model based Extensions of the EPC for Inter-Organisational Process Modelling

Christian Seel, Dominik Vanderhaeghen

Institute for Information Systems (IWi) at the
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, Geb. D3 2
D-66123 Saarbrücken
{ch.seel|vanderhaeghen}@iwi.uni-sb.de

Abstract: In the subject of information systems modelling languages have proven to be an effective mean to design business processes of enterprises and their information systems as well. These modelling languages are specified by meta-models. The choice of elements that are specified by the meta-model, which hence are part of the modelling language, is driven by the modelling languages' field of application. Areas of special interest for process modelling are inter-organisational collaboration scenarios. In this field, new requirements for business process models arise with the need to visualize new aspects of business processes. The review of existing modelling concepts is necessary to achieve an effective and efficient inter-organisational business process management. This paper motivates new requirements for collaborative business process models and presents a meta-model based extension Event-driven Process Chain (EPC), which can be considered as standard language for Business Process Management.

1 Introduction

During the last decades, a wide set of different information systems modelling languages has been applied for the design of enterprise processes and their supporting information systems. Modelling languages as, for instance the Petri-Net [Gr01], the Business Process Modeling Notation (BPMN) [Wh04], the Event-driven Process Chain (EPC) [KNS92], or language sets as the Unified Modeling Language (UML) [Oe03] ease the mapping of complex enterprise structures (data, processes, products and services, etc.) and behaviours (e.g. process flows, process costs) to a model-based image of the concerned reality. These modelling languages are usually specified by meta-models, e.g. the UML 2.0 is specified by use of the MOF [OMG05].

Due to changing fields of application a need for modelling language adjustment may arise because of an either less generic orientation or a too specific bias. **Collaborative Business (C-Business)** [SGZ03] [RS01] as a research field with the investigation of the emerging enterprise integration paradigm in collaboration scenarios has such influences on existing modelling languages and techniques: With a growing number of

collaborations and the combination of core competencies for a collaborative service or good production enterprises have to react concerning their design of enterprise business processes and their computer-assisted support on an inter-organisational level. The kind of and the quality of Business Process Management (BPM) activities, are depending on the modelling languages, techniques and tools which are applied to create an effective and efficient way to conduct C-Business. Hence, the influences of C-Business concerning the inter-organisational Business Process Management have to be analysed, which leads to the definition of requirements for the underlying methods and tools.

Thus, a state-of-the art reflection of process characteristics is provided in section 2. These characteristics have to be considered in modelling concepts and tools. For this purpose, we first evaluate the expressiveness of the Event-driven Process Chain (EPC) in collaborative scenarios, as one representative state-of-the art modelling language. To fill the gap between actual modelling concepts and to-be state we derive design recommendations for inter-organisational process modelling languages and apply them to the EPC. Therefore in section 4 a meta-model based extension of the EPC is presented.

2 Inter-Organisational Business Processes

As the most characterising part of C-Business, the processes among two or more enterprise departments, branches, but also business partners are integrated [Sc01a]. However, only sub-processes have been supported by BPM in the past. The single sub-processes of the collaborating enterprises must be integrated on different layers. The special process characteristics – the difference between usual, **intra**- and the new **inter**-organisational business processes – are mainly based on two pillars or field of interests as it is described in the following: first "cross-organisational business process flow" as a well-tangible, abstract object that describes static and dynamic process elements and characteristics and second "security and trust" as a fuzzy necessity for C-Business design, implementation and controlling.

Both classes of characteristics – coexisting interdependently and complementarily – are explained in the following. The subsequent derivation of requirements towards process modelling is described in section 2.2.

2.1 Characteristics of Inter-Organisational Business Processes

Business Process Flows

A business process includes different flows as a continuous, directed flow of data or information, goods and services or currency between process objects as well as the determination of the underlying processing logic. These flows remain inside an enterprise, e.g. as a data flow between departments (e.g. internal transfer of bills). However, they also cross enterprise borders and address external stakeholders as, for instance, it is with material goods- or immaterial service-flow (e.g. transport of goods

from a supplier via forwarding agent up to the manufacturer of the final product) [Hi98], [Th01], [Le03a]. A complex mesh of connections among the flows of individual, distributed process steps characterises those inter-organisational flows. From the view of one enterprise, internal process steps are initiated with well defined *interfaces* by external process events. Hence, we will have a closer look at these process interfaces implying flowing objects between different sending and receiving process participants.

This decentralisation leads to high coordination efforts of cross-enterprise business processes. The circumstance results in an increasing number of organisational and IT-related (process) interfaces [Sc01a]. Interfaces interlink activity-executing process objects such as application systems or databases, but also employees via data exchange as an information or control flow. Such interfaces may be identified both internally (intra-organisationally), within enterprises (e.g. between departments, branches, internal systems), and externally (inter-organisationally), between one enterprise and its business partners, customers, suppliers etc. [Sc03]. Such **process interfaces** cause friction losses. They result from, for example, long wait times caused by redundant work, high coordination efforts and competence splits [Th01]. In the following, we focus on external interfaces. They are divided into two main interface classes: process interfaces on a *human layer* and on a *technical, machine-oriented layer*.

Considering human beings as those actors who execute entrepreneurial activities manually we chose the term “organizational interface” for enterprise-spanning business processes that are characterised by interfaces concerning the organisational units involved in the process execution [Hi98]. The existence of organisational interfaces has numerous impacts on the way of process execution and on the involved organisational control instances. From an organisational point of view, explicit responsibilities concerning “power structures” and explicit “control instances” have to be defined in enterprise-spanning business processes. They ensure a smooth process flow even across the borders of a single enterprise [Sc02a]. However, this turns out to be particularly complicated in cross-enterprise processes as these responsibilities are mostly non-existent or even unintentional due to the kind of cooperation. A lack of organisational regulations leads to increasing coordination efforts concerning the execution of a process [Hi98].

Regarding mechanical actors information technology (IT) interfaces are identified. Cross-enterprise integration and optimisation of business processes mostly aims at a minimisation of interruptions in process flows caused by information and communication technology [Wö03]. In a cooperation scenario, the compatibility of application or information systems is required for a smooth cross-enterprise data and information exchange [Hi98]. The benefits concerning an increase in efficiency and effectiveness is only achieved with a vast implementation of processes over different application systems [Wö03]. Thus, the technical layer of process execution considering IT interfaces as relevant process interfaces are described in the following more detailed.

Heterogeneous IT landscapes may be identified as one central problem area causing ineffective and inefficient process hurdles. Predefined standards concerning existing IT interfaces have to ensure that IT systems used in the respective enterprises can be

integrated [Sc02a]. As various application systems of different manufacturers are mostly used in the different enterprises for heterogeneous purposes, the complexity of the process integration increases as a result of the heterogeneity of the IT landscapes. However, to reach business objectives, systems have to be interlinked through suitable mechanisms. Integration of heterogeneous application systems is mainly addressed within the field of Enterprise Application Integration (EAI) [Li00], however on a rather technical layer.

The number of existing **format mismatches**, which describes the frequency of changed communication media for the transmission of relevant information, may be seen as one resulting problem of IT heterogeneity. The manual transfer of information contained from a fax document (an order for instance) to an online form of an ERP system may serve as an example. In the context of inter-organisational business process controlling, the number of format mismatches can be used as a value for the analysis of the whole process [Bu03a].

The **use of different standards** may be identified as another relevant technical aspect within inter-organisational business processes. The integration of IT systems requires standardised methods for the connection of different communication end points and IT interfaces respectively. With heterogeneity of interfaces the integration effort increases [Wö03]. Inefficiencies concerning the electronic exchange of data and information can be eliminated by the definition of central semantic and syntactic standards for exchange objects (for example business documents) as well as transfer methods (transmission medium, exchange protocols etc.) [Mü03]. The complexity of a holistic process integration, caused by the multiplicity of potential business partners and IT systems to be integrated, is intensified by the existence of numerous, specific and partly very differently, sophisticated standardization approaches [Wö03]. This has a negative impact on the effort for process integration requiring adequate complexity reducing measures.

Moreover, the **automation** of process steps is also addressed with the help of IT as a primary objective in a cross-enterprise context. Here, a reduction of format mismatches is intended. With regard to process efficiency and effectiveness, improvements concerning process performance can be realized by the removal of manual process executions [Sc02a]. Furthermore, by defining automation as a value characteristic with Key-Performance-Indicators [SJ02], inter-organisational processes can be analysed in a measurable way [Bu03a].

Security and Trust

As a core part of business processes the exchange of information between employees is often problematic. Employees receive information willingly but they only reveal certain data under special circumstances [Wö03]. This aspect is even worse at the level of collaboration and inter-organisational business processes. The exchange of information is much more complicated due to cultural and mental aspects. Collaborations are characterized by insecurities during many phases of the collaboration life-cycle [We01]. With the use of the internet as the central medium for information exchange and transfer, the network economy turns out to be more impersonal and insecure in practice. The

electronic exchange of information is a weak point for every attack due to the described character of the medium internet. A disturbance of inter-organisational partner relationships may result. Thus, **security and trust** can be regarded as particularly critical for the inter-organisational process integration. Trust has to be regarded as an essential basis for an effective and efficient information exchange. It enables the communication between business partners. The described aspects have to be regarded as key success factors for the realisation of collaborative scenarios [Ra03].
Concerning the design of cross-enterprise processes, enterprises rely on negotiations for the coordination and the discussion or avoidance of potential partner conflicts. As the communication among the partners plays a major role in this context, **cultural** and **social discrepancies** have to be taken into account as well [Hi98]. Thereby, organisations are characterized considerably by cultural imprints because of organisational behaviour patterns and values [Sc97]. Within international cooperations, the problem to find common agreements due to cultural (e.g. language, terminology and understanding barriers, mental imprinting, legal distinctions) and temporal barriers (transcend time zones) by cooperating with acceptable coordination efforts is even aggravated [Wö03].

2.2 Modelling Requirements

Based upon the characteristics of section 2.1, requirements towards the modelling of business processes are defined in the following section to derive the language extensions presented in section 3 and 4.

From the main characteristic "**process interface"** which can be seen as a substitute for the necessity of process coordination the need for an explicit and purpose-driven description is derived. From a conceptual point of view, an adequate graphical visualisation is required. Furthermore, inter-organisational flows between human and mechanical process actors ought to be differentiated according to the kind of flow (see previous section). For a complete, cross-enterprise coordination additional information in the form of attributes may be necessary in a model as, e.g. pre- and post-conditions which limit the scope of interactions between the enterprise borders. Such attributes provide an execution-driven view on the necessary data for cross-enterprise business processes on a conceptual and even technical layer. Business processes may be coordinated with the aid of documentation models, but IT as for instance workflow engines or Enterprise Resource Planning (ERP) Systems might be also customized reverting to business process model information.

Moreover, the underlying **interaction points** with multiple end points [Bu03b] have to be specified by addressing the human layer with collaborating employees and departments on the one hand and the technical layer with IT on the other hand:

- Considering organisational units the area of authority has to be communicated to collaboration partners in order to ease the coordination of business processes. Without explicit responsibilities, business process objectives may be neglected, business process owners ought to be defined [Fr01] not only for small sub-processes but also on a cross-enterprise level. Detailed contact data ease the process execution through the minimisation

of coordination efforts. Transparency towards organisational structures with authority and communication relationships ought to be created.

- To derive appropriate and relevant information for the design of integrated but heterogeneous IT landscapes business process models have to be extended considering IT-related data. The crucial information which system covers which part of a sub-process has to be specified explicitly similarly to the description of organisational responsibilities. Moreover, interfaces ought to be described more in detail with special attributes marking heterogeneous systems (for instance syntactical description of exchange formats as e.g. individual XML structures, supported standards etc.) [Le03b] or necessary associations to related detail descriptions as sequence specification with UML sequence diagrams [Je04].

The modelling and the exchange of inter-organisational process model aiming at partner coordination have to be applied with reasonable efforts. Existing knowledge which is cast in models should be reused in order to save former investments [VZS05] of, e.g. Business Process Reengineering (BPR) - or Software Engineering (SE)-tasks. [Va05] Furthermore, a global sharing of information in the form of process models requires model integration and security mechanisms. Any flexible exchange of process data with heterogeneous description formats needs support in C-Business. Process data have to be secured and mechanism to hide critical information in models towards business partners need to be developed. [VZS05] Finally adaptation mechanisms for the translation of process models between different languages should be provided helping to overcome language-driven hurdles.

Summarizing, every possible weak point which may derange inter-organisational business process flows ought to be analysed in the design phase of the process, monitored during its implementation and controlled during the execution of enterprise-spanning business processes.

After this rather general requirements definition, precise description mechanisms are developed in the following sections considering the EPC as state-of-the art modelling language for BPM. In order to fulfil the requirements the extension of EPC becomes necessary. In the following chapter we hence introduce the EPC accomplishing the development of extensions due to the requirements as a practical example for the extension of modelling languages for inter-organisational process modelling. As languages are defined by their meta-models an extension of the EPC meta-model becomes necessary in order to allow the adequate representation of the inter-organisational process by the use of EPCs. Thus, beside the introduction towards EPC modelling the authors' definition of meta-modelling and then the meta-model of EPC is presented.

3 Business Process Modelling with Event-driven Process Chains

3.1 The Event-driven Process Chain and its Meta-Model

The **Event-driven Process Chain (EPC)** was developed in 1992 at the Institute for Information Systems in Saarbruecken in cooperation with SAP AG [KNS92]. EPC-models are central elements of BPM last but not least because of its use in the SAP R/3 reference model of SAP AG and the ARIS Toolset of IDS Scheer AG [Sc02b]. Enterprises model their process data as EPC-models in order to plan, design, simulate and control private enterprise processes. The EPC is a core part of the ARIS-framework and combines the different views towards the description of enterprises and information systems in the control view on the conceptual level. Few examples of EPC fragments, introducing the graphical EPC-visualisation, are given in section 4. The model elements of the EPC are introduced by the EPC's meta-model, which is presented in the following paragraph.

The term meta-model has its origin in the language levels of constructivism. This theory divides languages into languages on object-level and on meta-level [LS75]. The object language is the language, which is used to describe the objects of discourse. The meta language is the language, which describes the discourse itself. Transferred to modelling languages (cf. figure 1) a language oriented meta-model is model M2 which describes a language S1 that is used to create a model M1 of an object [Ho01]. STRAHINGER [St96] and HOLTEN [Ho00] divide a language into an ortho-language and a notation. The ortho-language describes all elements of modelling language unambiguous and without circles. The notation defines a graphical representation for each model element. One language can have multiple notations. Due to this definition the meta-model of a language and the notation has to be extended in order to create any adjustment.

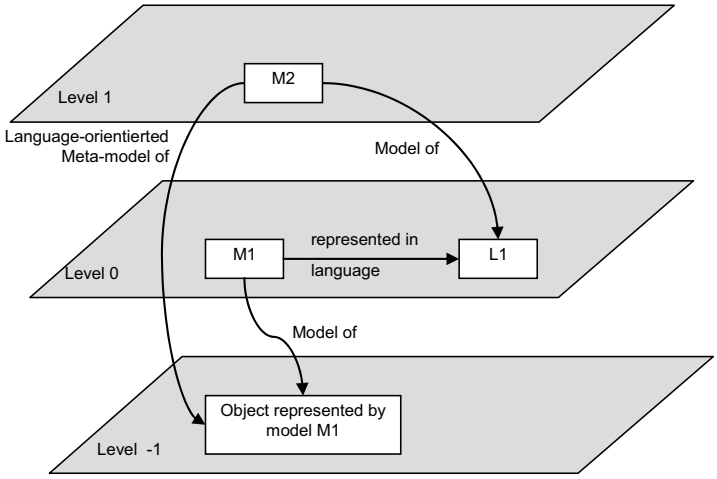

Fig. 1: Language-oriented Meta-Model [Ho01]

The EPC describes processes by the use of alternating functions and events as time-referring state changes Events und functions are linked by the control flow as directional edges [Ke00]. Functions describe activities and events passive states. Concerning the control flow functions and events can only be connected to each other. To split and join the control flow operator with the occurrences OR, XOR, and AND can be used after functions and events, except the OR- and XOR-Operators must not be used after events. The last remaining element, that could be connected via the control flow are process interfaces, which can be applied at the end and the beginning of an EPC to connect two EPCs from different models.

In the EPC meta-model (cf. fig. 2), which is constructed as an Entity-Relationship-Model (ERM) [Ch76], these four model elements are generalised as "process element". The connection between these four model elements are represented by the "Predecessor-Successor-Relationship". A problem of the generalisation to the "process element" and its recursive relationship are multiplicities of this relationship which should define which connections of these model elements are permitted. This problem can be solved by determining the multiplicities for each combination of predecessor and successor, e.g. if a function is followed by an event the multiplicities of the predecessor are (0,1) and the multiplicities of the successor are (0,1), too. The multiplicities of all possible cases are shown in Table 1.

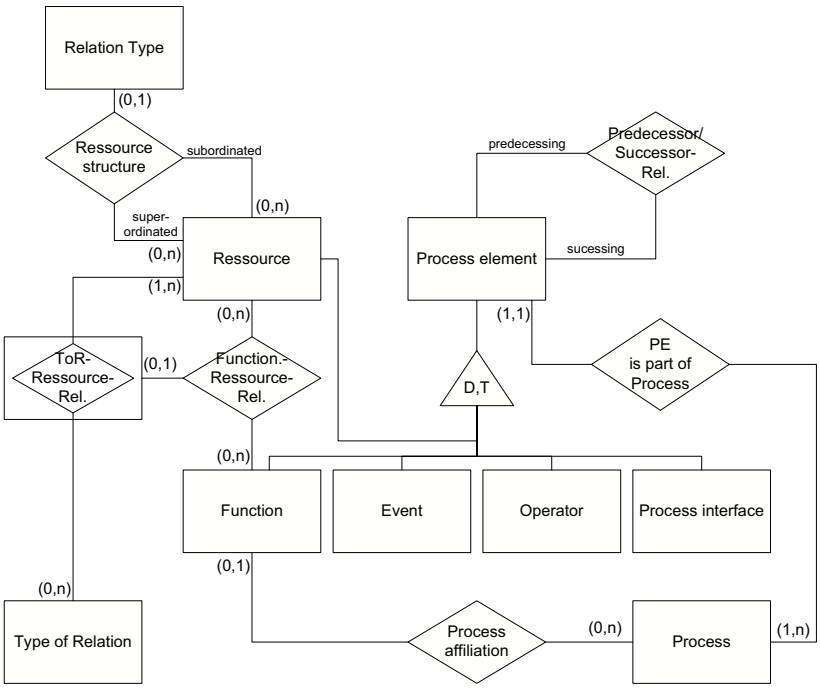

Fig. 2: Meta-Model of the eEPC

The next important meta-model element is the „process". Every process element is part of exactly one process and each process consists of one or more process elements. The

construct “process” can be used to create hierarchies of process models. Therefore a function is detailed by a sub-process. This refining of functions with sub-processes can be done over an unlimited number of levels. This issue is represented in the meta-model by the relation “process affiliation” between a function and a process, which allows, that one process can refine no or many functions. A function can either be not refined or be refined by exactly one process, because if a function was refined by more than one process the execution of the function would be ambiguous. The unlimited hierarchy levels of function affiliations are possible, because of the generalisation of the function to process element which is part of a process. So every sub-process consists of process elements, which could be function, that are refined by another sub process.

Table 1: Multiplicities of the Predecessor/Successor-Relationship, similar to [Be02]

Predecessor	Successor	Multiplicity Predecessor	Multiplicity Successor
Event	Function	(0,1)	(0,1)
Event	Event	(0,0)	(0,0)
Event	AND-Operator	(0,1)	(0,n)
Event	(X)OR-Operator	(0,1)	**(0,2..n)=(0,0) v. (2,n)**
Function/P.I.	Event	(0,1)	(0,1)
Function/P.I.	Function	(0,0)	(0,0)
Function/P.I.	AND-Operator	(0,1)	(0,n)
Function/P.I.	(X)OR-Operator	(0,1)	(0,n)
AND-Operator	Event	(0,n)	(0,1)
AND-Operator	Function	(0,n)	(0,1)
AND-Operator	AND-Operator	(0,n)	(0,n)
AND-Operator	(X)OR-Operator	(0,n)	(0,n)
(X)OR-Operator	Event	(0,n)	(0,1)
(X)OR-Operator	Function	(0,n)	(0,1)
(X)OR-Operator	AND-Operator	(0,n)	(0,n)
(X)OR-Operator	(X)OR-Operator	(0,n)	(0,n)
Starting Event	Process Interface	(0,1)	(1,1)
Process Interface	End Event	(1,1)	(0,1)
Event	Process Module	(0,1)	(0,1)
Function	Process Module	(0,0)	(0,0)
AND-Operator	Process Module	(0,0)	(0,0)
(X)OR-Operator	Process Module	(0,0)	(0,0)
Process Module	Event	(0,1)	(0,1)
Process Module	Function	(0,0)	(0,0)
Process Module	AND-Operator	(0,0)	(0,0)
Process Module	(X)OR-Operator	(0,0)	(0,0)

The last important characteristic of the modelling language EPC are resources, which can be annotated to functions. A resource[1] like, e.g. organisational units, applications systems or documents has its resource specific type relation to a function. For instance, an organisational unit can have the type of relation "is responsible for", which is not allowed for a document. So the meta-model contains a relationship between the type of relation and the resource, which determines which type of relation to a function is possible for what resource. As one resource can have more than one type of relation and one type of relation can be suitable for more than one resource, the relationship between a resource and a function is specified by a combination of the resource and its type of relation. Therefore, in the ER-Meta-Model the relationship "ToR-Ressource-Rel." is redefined to an entity type and creates a triple relationship with the entity type "function" and the entity type "resource". So the exact type of the relation for each connection of a resource to a function is specified. Additional resources can be related to each other, e.g. one organizational unit can have the relationship of the type "reports to" to another organisational unit. This is taken into account by the relationship "Resource Structure", which also includes the "type of relation".

To derive the need for the development of extensions, the modelling language is analysed due to the requirements defined in previous sections.

3.2 Evaluation for Inter-Organisational Business Process Modelling

The level of complexity escalates when trying to couple processes with each another in the development of a collaborative process model, as each network participant has their own "private" set of established methods (e.g. EPC, Petri-Net, UML Activity Diagram, BPMN) and tools (e.g. ARIS Toolset, VISIO, Rational Rose, eMagim, Metis) in use. Due to a lack of common interfaces and mapping-methods, neither can the tools interact with each other nor can the methods be transformed into one another.

The introduced EPC enables process modelling considering all relevant aspects for the description of business processes within the enterprise borders. Due to its connection to the ARIS framework, especially the ARIS House by SCHEER [Sc02c], an EPC model integrates the different views or perspectives on entrepreneurial entities. It builds up the dynamic connection with entity-relationships in the so called 'control view'. With the eEPC further elements such as process participants or data and information systems have been introduced. However, a clear focus on inter-organisational aspects does not yet exist.

Hence, we compare the capabilities of EPC modelling with the requirements described in section 2.2. This will be based upon the characteristics of processes in section 2.1. The eEPC, which was introduced above by its meta-model, enables holistic modelling of business processes considering control flows, organisational aspects, data entities as well

[1] There is no common understanding, which resources can be annotated to function in the EPC. An impression of the variety of resources gives the ARIS-Toolset of the IDS Scheer AG, which provides in its current release (6.2.3) 115 resource occurrences. Therefore, not all resources are regarded in detail in this paper.

as services and goods with every possible relationship in between. However, the eEPC does not differentiate the various kinds of process flows as connections between collaborating enterprises in a cross-organisational business process. An explicit visualisation of enterprise-spanning message triggers as it is introduced in the Business Process Modelling Notation (BPMN) with the so-called "Message Flow" is not envisaged. With this lack of information, EPC models loose expressiveness in the case of coordination. The explicit design or marking of the – for the execution of cross-organisational process responsible – process actors (human vs. machine) is not part of the eEPC. Enterprise-spanning interaction points are only included implicitly. Moreover, information which may become important for the coordination of process actors, are not part of the EPC modelling. Concerning human actors as the organisational units of an enterprise-network process-owners with their responsibilities are not explicit part of EPC processes. The scope of responsibilities is not shown in an EPC and may only be derived from the information which units act at a certain function. Coordination over the borders of an enterprise may become difficult because of such missing information in the business process models. Finally, even trivial attributes as contact data might be missing. This information is not required by the modelling language by default but might be implemented as a kind of model attribute in modelling tools. Aside from the human execution of process parts the technical dimensions should be specified in order to enable recognition of e.g. compatibility problems due to the need for heterogeneous application integration already at an early conceptual description level. Hence, the different kind of application systems must be visualized in an EPC with the crucial information of cross-organisational communication dependencies. While application systems themselves are already part of eEPCs, interaction points with a detailed description of data formats and flow descriptions (messages) are not contained. Hence, this should be introduced with, possibly, providing detailed data exchange information in an additional model as, e.g. UML sequence diagrams.

The mentioned requirements should be fulfilled by the modelling language itself in order to enable the application of the language in a special application domain. The simple addition of e.g. attributes depending on the use of a modelling tool complicates the modelling and integration of cross-organisational business process (parts) as few characteristics in modelling may differ due to the heterogeneity of tools.

4 Meta-Model based Extensions for Process Modelling

In the following section, extensions for the inter-organisational modelling of business processes are introduced. The propositions are based upon the requirements of section 2. They demonstrate a possible occurrence of the extensions considering the EPC.

Starting with the need for process interface visualisation we first want to create the ability to enable an identification of cross-enterprise process actors. Ordinary EPC models do not differentiate organisational units due to their affiliation to collaborating enterprises. Thereby, semantic faults as they might be caused through the use of homonyms and synonyms [RZ96], have to be avoided. Otherwise, misunderstandings as depict in Fig. 3 are possible. Affiliation should be described clearly expressing which

enterprise and which organisational unit or application system resp. process actor belongs together in an inter-organisational business process.

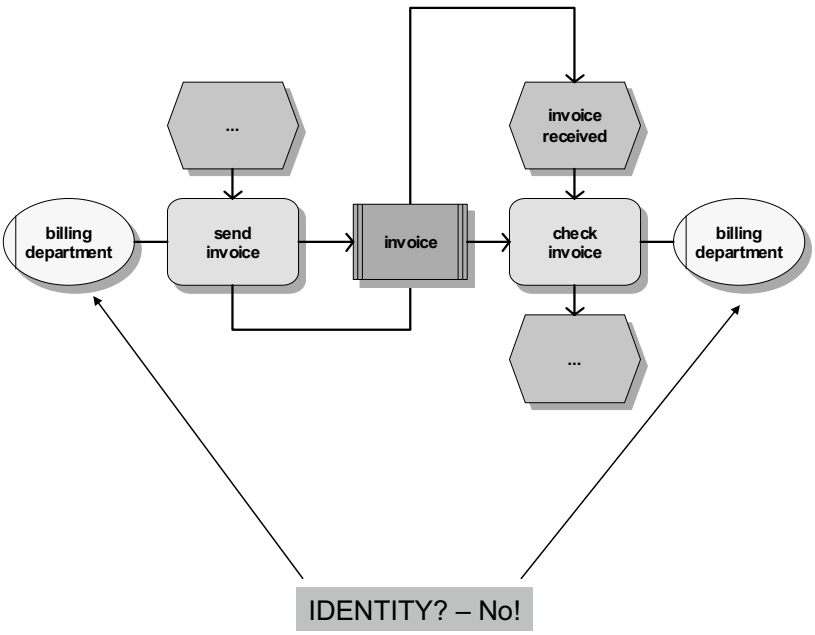

Fig. 3: Specification of cross-organisational process actors

As shown in Fig. 4 organisational groups depicting parts of a cross-organisational business process with the same affiliation are introduced. These groups might be completed with additional information due to the special modelling objective as, e.g. coordination of human process actors. In this case, adding contact partners as process owners to the organisational groups ease process coordination through the transparency of execution responsibilities.

In order to introduce the concepts of organisational groups to the EPC, a new model element "organisational group" has been added to the meta-model. As organisational groups consist of at least one process element, a relationship between the new meta-model element "organisational group" and the process element becomes necessary. The multiplicities of this relationship are (0,n) for the process element, because it must not belong to an organisational group, and (1,n) for the organizational group. To accommodate the idea that there is one contact person for each organisational group, from the meta-model element "organisational group" a relationship to the meta-model element "organisation unit", which is a specialisation of the resource, is added.

The grouping-approach aiming for an increase in transparency has already been introduced in former times with the column-concept of process chain diagrams [Sc94], or the pool and lane approach of, e.g. UML Activity Diagrams [Oe03] or the Business Process Modeling Notation [OR04], [Wh04]. Organisational interfaces are identified

easier. As a graphical occurrence small boxes enclosing relevant process parts might be the suitable notation for organisational groups.

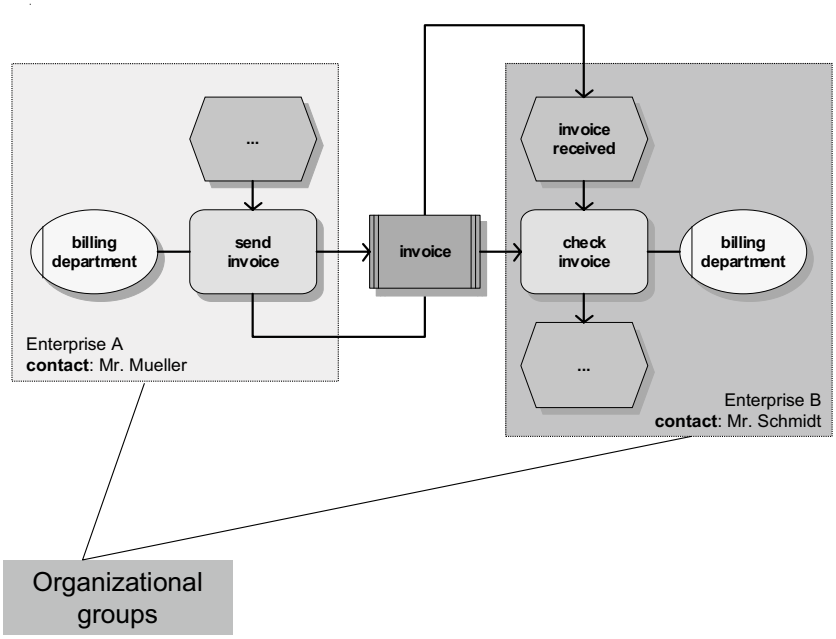

Fig. 4. Explicit differentiation of organisational groups in one process model

Similar to the grouping of organisational entities, interacting information systems could be marked within so called “application integration points”. Such interaction groups visualize process actors from a rather technical point of view. In an EPC, the group may be illustrated with a rectangle including all necessary interaction entities (application systems, interacting process steps, data exchange).

This interaction between process actors is realised via resources that are exchanged, e.g. documents or data objects. To introduce a model element in the EPC as described above in the meta-model, the relationship, which connects resources to functions, is redefined to an entity type and a recursive relationship with this redefined entity type is added. So it is possible in an EPC-model to create a new arch from one function of the first interacting process to the resource, that is used to interact, and from this resource to a function of the second interacting process, which directly indicates which functions interacts by the exchange of which resource.

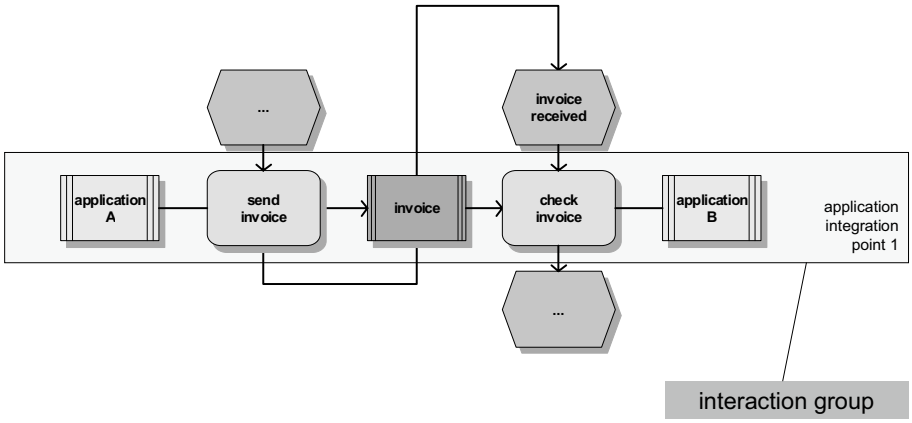

Fig. 5. Specification of heterogeneous enterprise application interactions

Aside from the rather less specified characterisation of application integration points, the exact description of system interfaces should be inserted to the EPC model due to the requirement of heterogeneous application integration coordination. Thus, detailed parameters as the syntax and semantic of EPC data clusters or conditions towards the execution of cross-organisational application interaction points (e.g. timer conditions, rules/constraints) might be shown in a detailed subgroup "interface description" of the interaction groups (cf. Fig. 6).

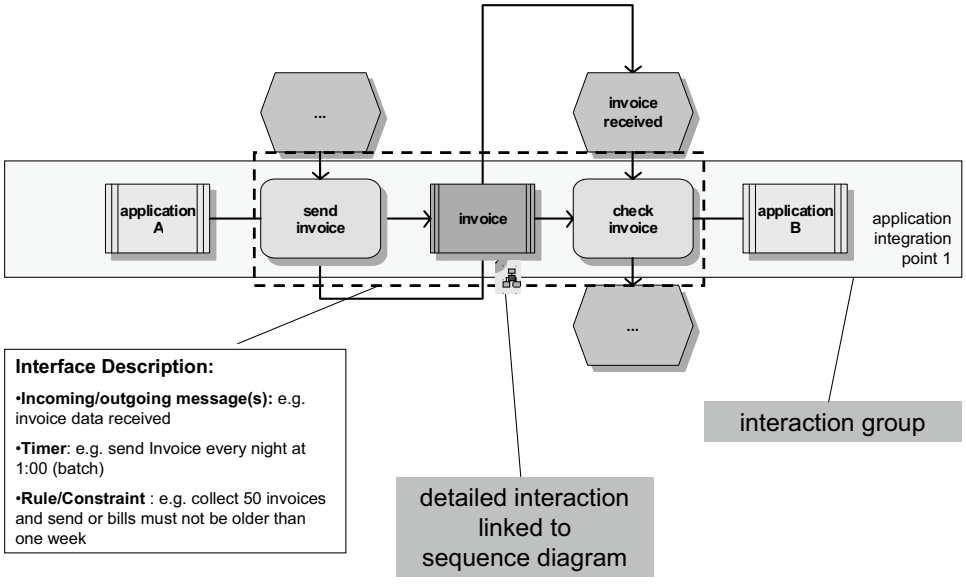

Fig. 6. Detailing enterprise application interfaces

The interface between two processes is an interface between two functions, because resources, like application systems or documents that link processes to each other can only be added to functions. So the interaction of processes is always realised by an interaction of functions. Therefore, the concept of "interface description" is integrated in the EPC as a recursive relationship of the meta-model element "function" with itself. But this relationship would only allow expressing that two functions are interacting, but not how they are interacting. So the relationship "interface description" is redefined to an entity type, which has two relationships. The first relationship is connected to the new model element "sequence diagram", which is a link to another model. The second relationship consists of the "interface description" and the "description object". The description object is only a model element on meta-model level and not supposed to be used in an EPC-Model, it is just a generalization for such model elements as "Timer", "rule/constraint" and "document/message". These three model elements represent the concept which has been described before.

Finally, security questions has been mentioned as an important issue restricting the scope of the modelling task and its result as a process model. To enable an exchange of process data using EPCs, information might be hidden as so called "process modules" [Ho05] [KKS04], which hide critical private process data. Only the information is shown in a process model which does not lead to economic disadvantages for single enterprises throughout the exchange of business process models. The rest is hidden behind substitute objects using abstraction mechanisms and reduction or extension. The kind of knowledge could be classified as *private*, public *partner* and *global* partner information. [VZS05], [Fr04]. The information towards view differentiation in EPC modelling has already been discussed in [LGB05], [Ad04].

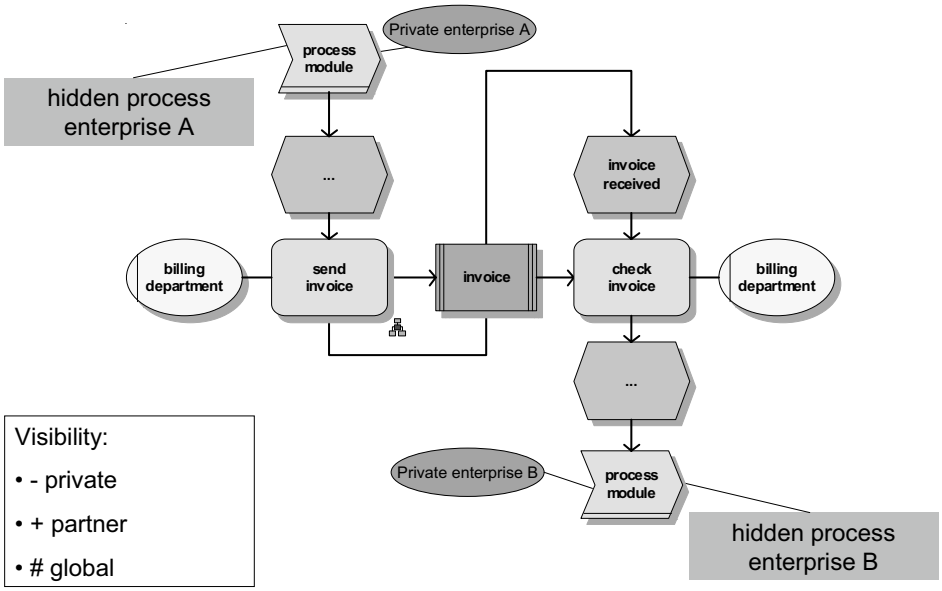

Fig. 7. Obtaining knowledge security through visibility access differentiation

In order to introduce this visibility approach to the EPC the grouping of model elements becomes necessary. Particularly as one process element could belong to more than one group, the new model element “process module” is added to the meta-model. It has a relationship “belongs to” with the meta-model element “process element”. Additionally, the “process element” has got a relationship to the meta-model element “visibility”, which has the occurrences “private”, “partner” and “global” according to the introduced differentiation. However, to specify the visibility for each “process module” is not enough, because if e.g. a private process module is visible depends on a particular perspective. So, for its owner a model element could be visible but for external partners, e.g., it wouldn’t. For this reason an attribute owner is added to the “process element”, which indicates the perspective towards visibility definition.

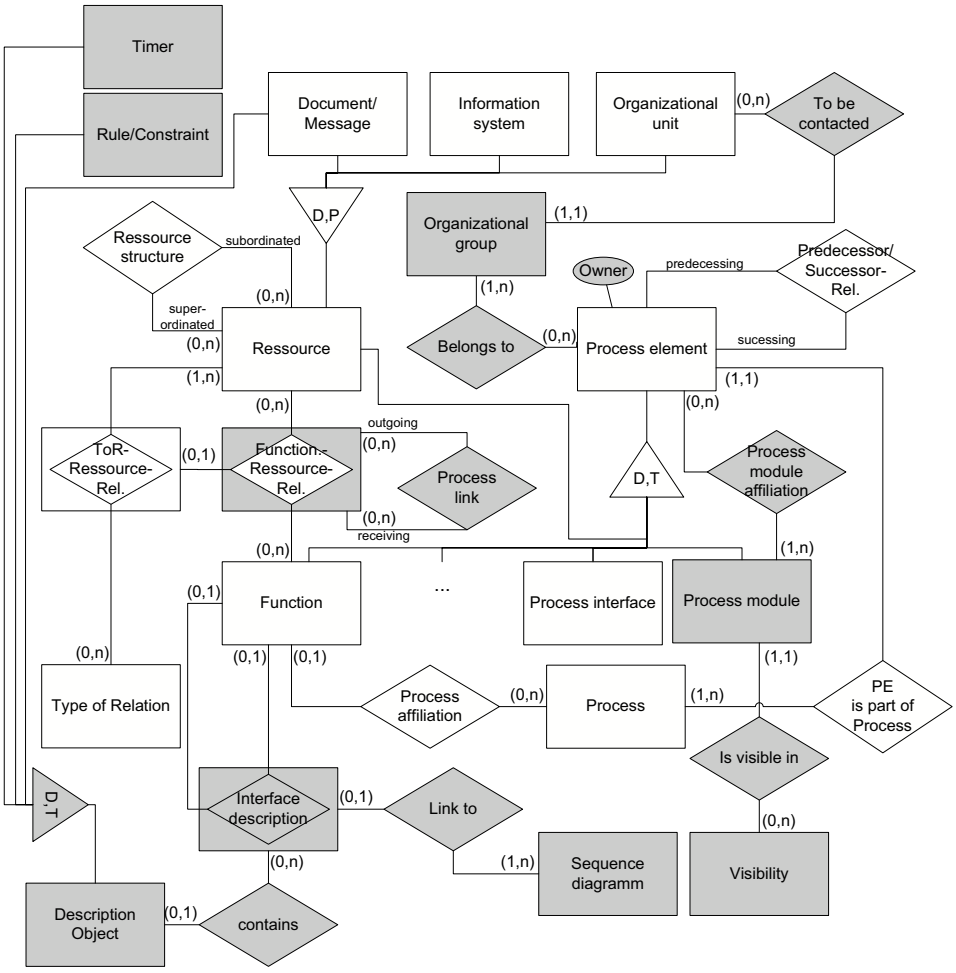

Fig. 8: Extended Meta-Model of the eEPC

Figure 8 shows the extended meta-model, which contains all necessary language elements to use the concepts for the EPC presented in the paper. The added meta-model elements are coloured in grey. The graphical representation of these added elements is not shown explicitly, because the examples above already present possible symbols.

5 Conclusions and Future Work

In the paper, we gave an overview of challenges for the modelling of business processes as a core part of Business Process Management. From the influences of a new application domain we derived requirements towards the scope of modelling and the modelling task itself. Moreover, a possible approach to fill the gap between as-is and to-be modelling with established modelling languages considering inter-organisational business processes has been presented. To demonstrate the requirements and their fulfilment an occurrence of the extensions for inter-organisational process modelling was derived for Event-driven Process Chains as a state-of-the art business process modelling language. Thereby, we identified problems in the cross-organisational EPC application and we proposed extensions for the EPC modelling.

The extensions for the language have been developed by the use of meta-models. Considering the method of meta-modelling to extend existing modelling languages, the presented meta-models have shown that ERM can appropriately be used to create meta-models and to extend languages. However, the expressiveness of an ERM is limited, concerning multiplicities in complex relationships. So in the field of meta-modelling, the development of a more expressive meta-modelling language is one of the future tasks.

However, few aspects considering inter-organisational modelling have not been solved: Social and cultural discrepancies as e.g. language barriers have to be addressed by flanking methods. Moreover, security- and trust-establishing actions need additional attention, so tool-based methods to hide process knowledge in defined collaboration scenarios according to an existing degree of partner confidence are a core part of future research. Finally, the presented extensions are going to be evaluated striving for a continuous improvement and a tool-based support.

The approaches – presented in the paper – have been developed at the Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence. The need for an appropriate inter-organisational business process modelling support is being addressed within the research project “ArKoS – Architecture for Collaborative Scenarios”[2]. Meta-modelling as a method to develop and to extent modelling languages is discussed within the research project “RefMod06 – Reference Modelling of conceptual Software Models for SMSE”[3]. Both projects are funded by the German Federal Ministry of Education and Research (BMBF).

---

[2] Further project information is available at http://www.arkos.info.
[3] Further project information is available at http://www.refmod06.de

References

[Ad04] Adam, O. et al.: Architecture for collaborative business process management – enabling dynamic collaboration. In: Dikbas, A.; Scherer, R. J. (Ed.): eWork and eBusiness in Architecture, Engineering and Construction. Leiden et al. A.A. Balkema Publishers, Istanbul, 2004, pp. 475-482.

[Be02] Becker, J.: Konfigurative Referenzmodellierung. In: Becker, J.; Knackstedt, R. (Ed.): Wissensmanagement mit Referenzmodellen. Springer, Heidelberg, 2002, pp. 25-144.

[Bu03a] Bullinger, H.-J. et al.: Unternehmensübergreifende Prozessintegration für den elektronischen Geschäftsverkehr. In: Kersten, W. (Ed.): E-Collaboration - Prozessoptimierung in der Wertschöpfungsketten. Dt. Univ.-Verl., Wiesbaden, 2003, pp. 59-82, p. 61, pp. 66-78, p. 68.

[Bu03b] Bussler, C.: B2B Integration - Concepts and Architecture. Springer, Berlin et al., 2003, pp. 29-30.

[Ch76] Chen, P.: The Entity-Relationship-Model : Toward a Unified View of Data. In: ACM Transactions on Database Systems 1 (1976), No. 1, pp. 9-36

[Fr01] Frank, U.: Organising the Corporation : Research Perspectives, Concepts and Diagram: Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 25.Universität Koblenz-Landau, Koblenz, 2001.

[Fr04] Frank, U. et al.: Business Process Definition Metamodel : concepts and overview.

[Gr01] Grob, H.-L.: Petri-Netz. In: Mertens, P. et al. (Eds.): Lexikon der Wirtschaftsinformatik. 4. eds. Springer, Berlin et al., 2001, pp. 369-370.

[Hi98] Hirschmann, P.: Kooperative Gestaltung unternehmensübergreifender Geschäftsprozesse. Gabler, Wiesbaden, 1998, pp.37-39, p. 66., pp. 69-70, pp. 77-80.

[Ho00] Holten, R.: Entwicklung einer Modellierungstechnik für Data-Warehouse-Fachkonzepte. In: Schmidt, H. (Ed.): Modellierung betrieblicher Informationssysteme : Proceedings der MobIS-Fachtagung 2000. Siegen, 2000, pp. 3-21.

[Ho01] Holten, R.: Metamodell. In: Mertens, P. (Ed.): Lexikon der Wirtschaftsinformatik. Springer, Berlin et al., 2001, pp. 300-301.

[Ho05] Hofer, A. et al.: Architektur zur Prozessinnovation in Wertschöpfungsnetzwerken. In: Scheer, A.-W. (Eds.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, No. 181, Universität des Saarlandes, Saarbrücken, 2005.

[Je04] Jeckle, M. et al.: UML 2 glasklar. Hanser, München et al., 2004, p. 323 et seq.

[Ke00] Keller, S.: Entwicklung einer Methode zur integrierten Modellierung von Strukturen und Prozessen in Produktionsunternehmen. VDI Verlag, Düsseldorf, 2000, p. 32 et seq.

[KKS04] Klein, R.; Kupsch, F.; Scheer, A.-W.: Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten. In: Scheer, A.-W. (Eds.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, No.178, Universität des Saarlandes, Saarbrücken, 2004.

[KNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozessmodellierung auf der Grundlage "ereignisgesteuerter Prozessketten (EPK)". In: Veröffentlichungen des Instituts für Wirtschaftsinformatik, No.89, Universität des Saarlandes, Saarbrücken, 1992, p. 2.

[Le03a] Leimstoll, U.: Kaved AG (Informing AG) - Elektroindustrie. In: Schubert, P.; Wölfle, R.; Dettling, W. (Eds.): E-Business Integration : Fallstudien zur Optimierung elektronischer Geschäftsprozesse. Hanser, München et al., 2003, pp. 67-80, p.73.

[Le03b] Lebender, M. et al.: Business Integration Software : Werkzeuge, Anbieter, Lösungen. In: Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO (Ed.): Fraunhofer IRB Verlag, 2003.

[LGB05] Lippe, S.; Greiner, U.; Barros, A.: A Survey on State of the Art to Facilitate Modelling of Cross-Organisational Business Processes. In: Nüttgens, M.; Mendling, J. (Eds.):

XML4BPM 2005, Proceedings of the 2nd GI Workshop XML4BPM -- XML for Business Process Management at 11th Conference Business, Technologie, and Web (BTW 2005) in Karlsruhe (Germany), 01 March 2005. Karlsruhe, 2005, pp. 7-21.

[Li00] Linthicum, D. S.: Enterprise application integration. 3. eds. Addison-Wesley, Boston et al, 2000.

[LS75] Lorenzen, P.; Schwemmer, O.: Konstruktive Logik, Ethik und Wissenschaftstheorie. 2. eds. Mannheim, 1975.

[Mü03] Müller-Lankenau, C.: Interessengemeinschaft Datenverbund für die Haustechnik IGH. In: Schubert, P.; Wölfle, R.; Dettling, W. (Eds.): E-Business Integration : Fallstudien zur Optimierung elektronischer Geschäftsprozesse. Hanser, München et al., 2003, pp. 123-136, p. 128.

[Oe03] Oestereich, B. et al.: Objektorientierte Geschäftsprozessmodellierung mit der UML. dpunkt, Heidelberg, 2003, pp.175-178.

[OMG05] Object Management Group: Meta-Object Facility, Version 1.4. URL http://www.omg.org/cgi-bin/apps/doc?formal/02-04-03.pdf [29.11.2005].

[OR04] Owen, M.; Raj, J.: BPMN and Business Process Management : Introduction to the New Business Process Modeling Standard. URL http://www.bpmi.org/bpmi-library/6AD5D16960.BPMN_and_BPM.pdf.

[Ra03] Ratnasingam, P.: Inter-Organizational Trust for Business-to-Business E-commerce. IBM Press, Hershey et al., 2003, pp. 2-4, p. 3.

[RS01] Röhricht, J.; Schlögel, C.: cBusiness : Erfolgreiche Internetstrategien durch collaborative business am Beispiel mySAP.com. Addison-Wesley, München et al., 2001, p.19.

[RZ96] Rosemann, M.; zur Mühlen, M.: Der Lösungsbeitrag von Metamodellen beim Vergleich von Workflowmanagementsystemen. In: Becker, J. et al. (Eds.): Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 48, Universität Münster, Münster, 1996, p. 10.

[Sc01a] Schmitt, R.: Unternehmensübergreifender Engineering Workflow : Verteilte Produktentwicklung auf der Grundlage eines parameterbasierten Daten- und Prozeßmanagements. Papierflieger, Clausthal-Zellerfeld, 2001, p. 37.

[Sc02a] Scheer, A.-W. et al.: Geschäftsprozessmanagement - The 2nd wave. In: IM Information Management & Consulting 17 (2002), Sonderausgabe, pp. 9-15, p. 11, p. 12.

[Sc02b] Scheer, A.-W.: ARIS: Von der Vision zur praktischen Geschäftsprozesssteuerung. In: Scheer, A.-W.; Jost, W. (Eds.): ARIS in der Praxis : Gestaltung, Implementierung und Optimierung von Geschäftsprozessen. Springer, Berlin et al., 2002.

[Sc02c] Scheer, A.-W.: ARIS - vom Geschäftsprozeß zum Anwendungssystem. 4. Eds., Springer, Berlin et al., 2002.

[Sc03] Schubert, P.: E-Business Integration. In: Schubert, P.; Wölfle, R.; Dettling, W. (Eds.): E-Business Integration : Fallstudien zur Optimierung elektronischer Geschäftsprozesse. Hanser, München et al., 2003, pp. 1-22, p. 5, p. 7.

[Sc94] Scheer, A.-W.: Business Process Engineering : Reference Models for Industrial Enterprises. 2. Eds. Springer, Berlin et al., 1994, p. 17.

[Sc97] Scholz, C.: Strategische Organisation : Prinzipien zur Vitalisierung und Virtualisierung. Moderne Industrie , Landsberg/Lech, 1997, pp. 225-229.

[SGZ03] Scheer, A.-W.; Grieble, O.; Zang, S.: Collaborative Business Management. In: Kersten, W. (Ed.): E-Collaboration : Prozessoptimierung in der Wertschöpfungskette. Dt. Univ.-Verl., Wiesbaden, 2003, pp. 29-57, p. 42;

[SJ02] Scheer, A.-W.; Jost, W.: Geschäftsprozessmanagement : Kernaufgabe einer jeden Unternehmensorganisation. In: Scheer, A.-W.; Jost, W. (Eds.): ARIS in der Praxis : Gestaltung, Implementierung und Optimierung von Geschäftsprozessen. Springer, Berlin et al., 2002, pp. 33-44, p 43.

[St96] Strahinger, S.: Metamodellierung als Instrument des Methodenvergleichs : Eine Evaluierung am Beispiel objetktorientierter Analysemethoden. Shaker, Aachen, 1996.

[Th01] Thaler, K.: Supply Chain Management : Prozessoptimierung in der logistischen Kette. 3. eds. Fortis Verlag, Köln et al., 2001, pp. 43-44, p. 46, p. 49, p.76, p. 95, p. 117.

[Va05] Vanderhaeghen, D. et al.: XML-based Transformation of Business Process Models - Enabler for Collaborative Business Process Management. In: Nüttgens, M.; Mendling, J. (Eds.): XML4BPM 2005, Proceedings of the 2nd GI Workshop XML4BPM -- XML for Business Process Management at 11th Conference Business, Technologie, and Web (BTW 2005) in Karlsruhe (Germany), 01 March 2005. Karlsruhe, 2005, pp. 81-94.

[VZS05] Vanderhaeghen, D.; Zang, S.; Scheer, A.-W.: Interorganisationales Geschäftsprozessmanagement durch Modelltransformation. In: Scheer, A.-W. (Eds.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, No. 182, Universität des Saarlandes, Saarbrücken, 2005, pp. 6-7.

[We01] Wehner, J.: Projektnetzwerke - Neue Unternehmensstrukturen und neue Qualifizierungen. In: Rohde, M.; Rittenbruch, W.; Wulf, V. (Eds.): Auf dem Weg zur virtuellen Organisation : Fallstudien, Problembeschreibungen, Lösungskonzepte. Physica-Verlag, Heidelberg, 2001, pp. 33-54, p. 41.

[Wh04] White, S. A.: Business Process Modeling Notation (BPMN). URL http://www.bpmi.org/bpmi-downloads/BPMN-V1.0.pdf.

[Wö03] Wölfle, R.: Stellenwert von E-Business-Integrationsprojekten in Unternehmen. In: Schubert, P.; Wölfle, R.; Dettling, W. (Eds.): E-Business Integration : Fallstudien zur Optimierung elektronischer Geschäftsprozesse. Hanser, München et al., 2003, pp. 23-38, p. 29, p. 27, p. 34, p. 37.

Modellierung von Flexibilität mit Ereignisgesteuerten Prozessketten (EPK)

Jürgen Grief*, Heinrich Seidlmeier**

*BC & S GmbH
Bank Consulting & Solutions
Schumanstr. 33
52146 Würselen
J.Grief@bcs-gmbh.de

**Fachhochschule Rosenheim
Institut für Organisation und Wirtschaftsinformatik
am Fachbereich Betriebswirtschaft
Hochschulstr. 1
83024 Rosenheim
seidlmeier@fh-rosenheim.de

Abstract: „Flexibilität“ oder auch „Adaptivität“ ist, gleichermaßen in Theorie und Praxis, in Zeiten der beschleunigten Ökonomie einer der betrieblichen Erfolgsfaktoren schlechthin. Diese Eigenschaft eines Unternehmens, das dann als „agil“ gilt, wird auch beim Management von Geschäftsprozessen gefordert. Auf der strategischen Ebene finden sich einschlägige Hinweise in der wissenschaftlichen Literatur, in Marketingprospekten oder in Firmenbroschüren in großer Zahl. Auf der „tiefen“ Ebene der detaillierten Prozessmodellierung zeigt sich ein anderes Bild. Ausführungen zur konkreten Modellierung von flexiblen Prozessen sind eher selten. Hier setzt dieser Beitrag an. Es werden Konstruktionsideen für flexible Prozesse entwickelt und mit EPK umgesetzt. Die grundlegenden Ideen „Modularisierung“ und „Entkopplung“ führen zur selbststeuernden Prozesskonfiguration und variablen Funktionsreihenfolge. Damit wird Flexibilität zwischen und in entkoppelten Prozessmodulen erzeugt.

1 Bedeutung und Definition von Flexibilität

Schon seit Jahren unstrittig ist sicherlich die Bedeutung der Flexibilität für den Unternehmenserfolg [z.B. GSW98], insbesondere in kleineren und mittelständischen Unternehmen [z.B. Fe86]. Unter anderem wird die Fähigkeit flexibel (oder auch adaptiv bzw. agil) zu sein, als das wichtigste Ziel des Geschäftsprozessmanagements erachtet [Jo05, S. 9]. Weiterhin ergab eine Umfrage unter mehr als 4000 Führungskräften weltweit, dass die Fähigkeit, Flexibilität tatsächlich auch in Unternehmenswandel umzusetzen, als größte Managementherausforderung bis 2010 betrachtet wird [Ec2005].

Insbesondere ist beispielsweise im Bankensektor durch die rapide zunehmenden Marktanteile der Transaktionsbanken der Bedarf an flexiblen Referenzprozessen deutlich gewachsen. Die jeweilige Transaktionsbank definiert einen Referenzprozess, welcher dann mandantenspezifisch anzupassen ist. Diese flexiblen Referenzprozesse stellen einen wesentlichen Faktor für die Marktposition der Transaktionsbanken dar. Der Erfolg der Transaktionsbanken legt den Schluss nahe, dass in naher Zukunft weitere „Dienstleistungsfabriken" in anderen Geschäftsfeldern entstehen werden, und somit die Bedeutung flexibler (Referenz-)Prozesse zunehmend steigen wird.

Trotz der hier nur angerissenen Wichtigkeit wird die tiefere Auseinandersetzung mit dem vielgenannten Phänomen „Flexibilität" oft vernachlässigt, gerade auch auf Prozessmodellierungsebene. Die eigentlich zuständige klassische (deutschsprachige) Organisationsliteratur setzt sich nur sehr oberflächlich damit auseinander [z.B.KK92 und Sc03], ebenso die älteren und gleichermaßen neueren Prozessmanagement-Standardwerke [z.B. HC94 und BKR05].

Flexibilität wird sehr unterschiedlich definiert. Generell kann man unter Flexibilität (im Sinne von Adaptivität) die Fähigkeit verstehen, sich ändernden Rahmenbedingungen anpassen zu können [Al98, S. 2]. Mit Betonung des Zeitfaktors ist ein System dann flexibel, „wenn einem Wandlungsbedarf ein in angemessener Zeit aktivierbares Wandlungspotential im System gegenübersteht" [AS04, S. 70]; Flexibilität ist folglich als Wandlungsfähigkeit zu verstehen. Neben dem Zeitfaktor ist ferner der Anpassungsaufwand zu berücksichtigen. Ein Prozess ist nur dann flexibel, wenn ein geringer Wandlungsbedarf nicht die Aktivierung eines diesem Bedarf unangemessenen Wandlungspotenzials erfordert.

Diese Anpassungs- bzw. Wandlungsfähigkeit wird auch diesem Beitrag zugrunde gelegt. Vor diesem Hintergrund werden im folgenden Kapitel verschiedene allgemeingültige Ansätze zur flexiblen Gestaltung von Prozessen kurz dargestellt. Es wird sich zeigen, dass alle Vorschläge im Kern auf wenigen gemeinsamen Grundideen aufbauen. Diese Kerngedanken werden im darauf folgenden Kapitel aufgegriffen, weiterentwickelt und als Basis für flexible Ereignisgesteuerte Prozessketten verwendet.

2 Allgemeine Ansätze zur Bewältigung von Umweltänderungen in Prozessen

Es existiert eine ganze Reihe von Möglichkeiten, um den Anforderungen gerecht zu werden, die aus Umweltänderungen resultieren. Nachfolgend werden einige davon kurz dargestellt.

2.1 Erzeugung von Prozessvarianten

Übersichtsartig berichtet Allweyer über „Prozesshandbücher“, „Prozessbibliotheken“ und „Prozesspartikel“ [Al98, S. 67 f. und die dort ausgewertete Literatur]. In einem Prozesshandbuch werden Teilprozesse und Regeln für deren Verknüpfung abgelegt. Auf dieser Basis lassen sich, wenn Wandlungsbedarf entsteht, Prozessalternativen erzeugen. Allerdings gibt es keine Angaben, welche Alternativen zielgerecht unter den gegebenen Rahmenbedingungen zu wählen sind. Eine Prozessbibliothek enthält wieder verwendbare Referenzprozessbausteine. Regeln bzw. Kriterien zur Ermittlung der geeigneten Bausteine werden im Einzelnen nicht beschrieben. Im Rahmen der Verwendung von Prozesspartikeln zur Prozessgestaltung werden die Anwendungsvoraussetzungen für die Partikel ausdrücklich behandelt. Zentral ist ein feststehendes, so genanntes „Essentielle Modell“ mit den zur Zielerreichung unbedingt notwendigen Prozessschritten und Abhängigkeiten. Aufgrund von Umweltveränderungen notwendige Prozessvarianten werden auf Basis des essentiellen Modells und passender generischer Prozesspartikel modelliert. Da die Einsatzvoraussetzungen von Prozesspartikeln definiert sind, können geeignete Prozessalternativen ausgewählt werden.

Da in allen genannten Fällen (Handbuch, Bibliothek, Partikel) letztlich Prozessvarianten erzeugt werden, spielt das „Management“ dieser Varianten, bevorzugt durch entsprechende Modellierungswerkzeuge, eine wichtige Rolle.

Die effiziente Handhabung von Prozessvarianten stellt eine wesentliche Grundlage für den Erfolg von Referenzprozessen dar. Ausgehend von einem branchenspezifischen Referenzprozess wird ein unternehmensspezifischer Prozess generiert, welcher allgemein als Prozessvariante zu interpretieren ist. Führt eine Dienstleistungsfabrik (Beispiel: Transaktionsbank) die Prozesse mehrerer Unternehmen (derselben Branche) durch, so muss der Referenzprozess derart flexibel gestaltet sein, dass aus ihm mit vertretbarem Aufwand unternehmensspezifische Prozesse abgeleitet und diese mit ihm synchron gehalten werden können.

2.2 Verwendung „robuster“ Modellierungsmethoden

Weitere Vorschläge zur prozessorientierten Bewältigung von Umweltänderungen sind die objektorientierte und die ressourcenbasierte Modellierung.

Die Wurzeln der objektorientierten Softwareentwicklung reichen bis in die 1970er Jahre zurück. Ab den 90er Jahren setzt sich dieses Paradigma zunehmend gegen die prozedurale Vorgehensweise durch [Oe2003, S. 12]. Die objektorientierte Modellierung von Prozessen, insbesondere mit der Unified Modeling Language (UML), ist in etwa ab Mitte der 90er bekannt [Zi99, S. 65 ff.]. Weiterhin existieren Vorschläge zu „Objektorientierten Ereignisgesteuerten Prozessketten“ (oEPK) [SNZ97], auch in Kombination mit der UML [LA98; Da99].

Gerade durch die Zusammenführung von objektorientierter, eher umsetzungstechnischer („UML“) und prozessorientierter, eher betriebswirtschaftlich-konzeptioneller Sichtweisen („EPK“) verspricht man sich, die Lücken und Schwächen des jeweils einen Ansatzes durch die Integration mit dem jeweils anderen Ansatz zu umgehen. Eine Gegenüberstellung der wesentlichen Elemente von UML Aktivitätsdiagrammen und EPK findet sich in [Gr05].

Die an Unternehmensressourcen ausgerichtete Prozessmodellierung ist eine sehr neue Sichtweise [BMH05; Se05]. Der „Resource-Based View of the Firm“ führt den Erfolg eines Unternehmens auf Ressourcenasymmetrien zurück. Dauerhaft überdurchschnittliche Gewinne und nachhaltige, verteidigungsfähige strategische Wettbewerbsvorteile als deren Ursache erklären sich demnach aus einer überlegenen Ressourcenausstattung und/oder –nutzung [grundlegend Ba91].

Prozesse werden dabei als Aneinanderreihung von ressourcenbasierten Prozessbausteinen betrachtet. Ein Prozessbaustein besteht aus einer Funktion (= Prozessschritt), dazu notwendigen Ressourcen (beispielsweise Sachmittel) und Fähigkeiten zur Bewältigung der Funktionsdurchführung. Auf der Grundlage dieser vorhandenen Bausteine können dann Prozesse nach (Wandlungs-) Bedarf zusammengestellt werden.

Da in beiden Fällen mit „Objekten“ und „Ressourcen“ eher robuste Prozessbestandteile[1] im Vordergrund stehen, sind darauf aufbauende Modelle weniger sensibel gegenüber Umweltänderungen und können ohne Modifikationen weiterverwendet werden. Derartige Prozesse sind damit nicht im eigentlichen Sinne flexibel, können aber bevorzugt in dynamischen Umwelten eingesetzt werden.

2.3 Wahl des geeigneten Abstraktionsgrades

Unter Abstraktionsgrad soll die Detaillierung, auch Granularität der Prozessmodellierung verstanden werden. Es handelt sich nicht um einen eigenständigen Ansatz, sondern um ein grundlegendes Konzept. Allgemein gilt: Je feiner ein Prozess modelliert ist, desto genauer ist sein Ablauf festgelegt. Damit fehlen Freiheitsgrade in der Abarbeitung und reduzieren als Folge die in Kapitel 1 angesprochenen Wandlungspotentiale. Für sehr abstrakt formulierte Prozesse gilt das Gegenteil.

Besteht beispielsweise der Prozess „Rechnung bearbeiten“ nur aus einem, gleichnamigen Prozessschritt, kann die Bearbeitung einer Rechnung sehr flexibel erfolgen – im Bedarfsfall z.B. weniger genau, bevorzugt oder beschleunigt. Zur eigentlichen Aufgabe „Rechnung bearbeiten“ existiert keine weitergehende Bearbeitungsanweisung. Eine sehr genaue Ablaufbeschreibung ver- oder zumindest behindert flexible Vorgehensweisen. Von zentraler Bedeutung ist die Bestimmung des „optimalen“ Abstraktionsgrades.

---

[1] Vgl. zur Prozesseigenschaft „Robustheit“ [Al98, S. 89 ff.].

Damit im Falle von sehr groben Prozessbeschreibungen vorgegebene Prozessziele trotzdem erreicht werden (Flexibilität damit nicht opportunistisch ausgenutzt wird), bieten sich verschiedene organisatorische Lösungen an [PDF05, S. 244 f.]. Z.B. kann bei der Auswahl der entsprechenden Mitarbeiter auf das Vorhandensein notwendiger Werte (Genauigkeit, Loyalität, Zuverlässigkeit u.ä.) geachtet werden.

Die Entscheidungsmöglichkeiten, die eine grob-granulare Prozessbeschreibung hinsichtlich der manuellen Durchführung eines Prozesses bietet, setzen Kenntnisse bezüglich möglicher Aktionen und Effekte voraus. Dieses „Prozesswissen" des Ausführenden muss auch in die Beschreibung automatischer Prozesse integriert werden, damit ausführende Systeme in analoger Weise reagieren können.

2.4 Zusammenfassende Erkenntnisse

Zunächst fällt auf, dass sehr viele der in den vorigen Abschnitten genannten Ansätze mit Teilprozessen, Prozessbausteinen u.ä. sowie sich daraus ergebenden Varianten arbeiten. Allerdings ist die Bildung von Teilprozessen, und damit verbunden auch die Wahl des optimalen Abstraktionsgrades, nicht durchgängig gelöst. Auch die richtige Verknüpfung der Teilprozesse ist nicht in allen Fällen klar. Diese methodische Lücke ist bei der zielgerechten Wahl von Alternativen, unter den gegebenen Rahmenbedingungen, sogar noch größer. Interessant erscheint, weil in den beschriebenen Ansätzen vernachlässigt, die methodische Überlegung, aus Teilprozessen automatisch Gesamtprozesse zu erzeugen.

3 Flexible Prozesse durch Modularisierung und Entkopplung

Der in diesem Absatz dargestellten Konstruktion flexibler Prozesse, formuliert als (weitergehend zu diskutierende) „Konstruktionsideen", liegt folgende Basisüberlegung zu Grunde. Flexibilität in Organisationen wird generell durch Modularisierung von organisatorischen Strukturen und Prozessen sowie durch die Entkopplung dieser Module erreicht. Entkoppelte Module erleichtern Neukonfigurationen von Strukturen und Prozessen und erhöhen somit die organisatorische Flexibilität [ähnlich AS04, S. 70 f.].[2] Die konkrete Umsetzung dieser Ideen mit EPK erfolgt im folgenden Kapitel 4.

[2] Diese auf organisatorische Aspekte ausgerichtete Sichtweise kommt grundsätzlich auch zur Anwendung bei der Konstruktion flexibler Softwarearchitekturen [SK03].

Konstruktionsidee 1: Modularisierung

Als erste grundlegende Idee wird vorgeschlagen, Prozesse aus Modulen aufzubauen. Ein Modul besteht aus Schnittstellen und einem Kern. Der Kern setzt sich aus grundsätzlich freien Bausteinen mit der eigentlichen Funktionalität[3] zusammen. Diese Module können auch als Ressourcen [Se05] oder Services [BMH05] verstanden werden.

Konstruktionsidee 2: Entkopplung der Module

Die Flexibilisierung von Prozessen und damit die Erzeugung von Wandlungspotential werden durch die Entkopplung der Module erreicht. Es entstehen, als zweite wesentliche Idee, autonome Prozessmodule (bzw. Teilprozesse), die situativ, aber zielgerecht zusammengesetzt werden können.

Konstruktionsidee 3: Modul-Intra- und –Interflexibilität

Modularisierung und Entkopplung erzeugen Flexibilität auf zwei Ebenen:

- Innerhalb der Module („Modul-Intraflexibilität"): Die freien (bzw. auch entkoppelten) Prozessbausteine können in einem Modul grundsätzlich beliebig kombiniert werden.

- Zwischen den Modulen („Modul-Interflexibilität"): Die entkoppelten Prozessmodule bzw. Teilprozesse können grundsätzlich beliebig zu einem Gesamtprozess kombiniert werden.

Konstruktionsidee 4: Selbststeuernde Prozesskonfiguration

Ein „Kopplungssystem" enthält die Methodik und die notwendigen Umweltinformationen, um die Prozessmodule selbständig zu Gesamtprozessen zusammenbauen zu können. Diese Idee reduziert die durch Flexibilität gewöhnlich erzeugte Komplexität. Es müssen nicht alle in der Diskurswelt denkbaren Modulkombinationen (als Redundanz erzeugende Varianten) vorgehalten werden – mit der Gefahr einer „kombinatorischen Explosion". Der von nicht beeinflussbaren Umweltveränderungen ausgelöste Wandlungsbedarf wird durch die selbständige Erzeugung der einen passenden Prozessvariante gedeckt. Das Wandlungspotential ist nicht explizit modelliert, sondern liegt v.a. im modul-interflexiblen Kopplungssystem.[4]

---

[3] Vgl. zu den Gestaltungszielen bei der Modularisierung und zum Grad der Modularisierbarkeit [AS04 S. 71f. und S. 74]. Gemäß der ARIS-Methodik [Sc01] setzt sich ein Baustein aus den Sichten Funktion, Organisation, Daten und Leistung zusammen.

[4] Und daneben in der Modul-Intraflexibilität.

Zur methodischen Umsetzung dieser Ideen bieten sich die Ereignisse in EPK an. Zum einen bilden Ereignisse Umweltveränderungen ab. Eine derartige Wandlungsbedarf erzeugende Änderung ist nichts anderes als ein Ereignis. Zum anderen werden nachfolgend Ereignisse zur Umsetzung des oben angesprochenen Kopplungssystems in Form von (ARIS-) Ereignisdiagrammen herangezogen.

Konstruktionsidee 5: „Kernmodell“

Das Kernmodell enthält zwingend notwendige Prozessmodule und schränkt dadurch die Flexibilität bei der Prozessgestaltung ein. Die Notwendigkeit eines derartigen Kernmodells kann sich aus fixierten Unternehmensvorgaben bzw. Prozesszielen (z.B. „100%-Endkontrolle“) oder aus vorhandenen rechtlichen, betrieblichen oder sonstigen Verordnungen (z.B. „Vier-Augen-Prinzip“) ergeben. Weiterhin erzeugt die Verwendung eines Kernmodells „kontrollierte“ Flexibilität, indem es ein „Ausufern“ (z.B. übermäßiges bzw. unwirtschaftliches Erfüllen von Kundenänderungswünschen) bzw. sogar im Extremfall Willkür verhindert.

4 Umsetzung der Konstruktionsideen mit EPK

Die Konstruktionsidee der selbststeuernden Prozesskonfiguration wird im ersten Teil dieses Kapitels auf der Basis von EPK und Ereignisdiagrammen umgesetzt. Die auf diesem Wege erzielte Flexibilisierung von Prozessdarstellungen führt zu einer signifikanten Verbesserung der Modul-Interflexibilität.

Im Anschluss wird durch ein Konzept zur Abbildung variabler Funktionsreihenfolgen innerhalb von EPK ein Ansatz zur Erhöhung der Modul-Intraflexibilität vorgestellt.

4.1 Selbststeuernde Prozesskonfiguration

In der EPK-Literatur wird der Ereignisbegriff sehr weit gefasst. Chen und Scheer definieren in dem Grundsatzartikel [CS94] Ereignisse wie folgt: „Ereignisse in der EPK lösen Funktionen aus und sind deren Ergebnis. Sie repräsentieren zugleich das Ende einer Funktionsausführung und den Ausführungsbeginn der Nachfolger-Funktion. […]“. Dieser Ereignisbegriff entspricht in seinem Kern auch der Definition in [KNS92]. Rump verwendet zwar in [Ru99] den Ereignisbegriff aus [KNS92], er weist jedoch darauf hin, dass er ihn letztlich im Sinne einer Zustandsdefinition nutzt. Davis folgt dem Ereignisbegriff aus [KNS92] in [Da03] weitgehend, wobei er zwischen Startereignissen, mittleren Ereignissen und Endereignissen unterscheidet.

Uthmann geht in seiner Ereignisdefinition in [Ut97] einen Schritt weiter: „Ereignisse bezeichnen Zustandsübergänge, wobei Bereitstellungs- und Auslöseereignisse unterschieden werden, die sich jeweils auf die Bereitstellung von Objekten als Ergebnis einer ausgeführten Funktion bzw. auf das Eintreten einer Objektkonstellation beziehen, die zur Auslösung einer Funktion führt. In der Konsequenz beginnt jede Prozeßkette mit einem oder mehreren Auslöseereignissen, die eine prozeßinstanziierende Anfangsbedingung bzw. mehrere -teilbedingungen repräsentieren, und endet mit einem oder mehreren Bereitstellungsereignissen, die das Eintreten des Zustandes nach Prozeßende bezeichnen.“.

Während Uthmann lediglich Bereitstellungs- und Auslöseereignisse unterscheidet[5], werden wir eine Kategorisierung in drei verschiedene Ereignistypen vornehmen, auf deren Grundlage die selbststeuernde Prozesskonfiguration umgesetzt wird. Auf der Basis des Ereignisbegriffs aus [CS94] seien folgende Ereignistypen definiert:

- Ein Bereitstellungsereignis ist ein Ereignis, das unmittelbar aus einer Funktionsausführung resultiert.

- Ein internes Auslöseereignis ist ein Ereignis, welches im Rahmen des Prozessablaufs die unmittelbare Konsequenz von Bereitstellungsereignissen oder externen Auslöseereignissen (oder Kombinationen dieser Ereignisse) ist, und – ggf. in Verbindung mit weiteren (internen oder externen) Auslöseereignissen – zu der Ausführung einer oder mehrerer Funktionen führt.

- Ein externes Auslöseereignis ist ein Ereignis, welches nicht aus dem Prozessverlauf resultiert, sondern außerhalb des betrachteten Prozesses ausgelöst worden ist, und – ggf. in Kombination mit weiteren (internen oder externen) Auslöseereignissen – zu der Ausführung einer oder mehrerer Funktionen führt.

Die strikte Unterscheidung von Bereitstellungs- und Auslöseereignissen ermöglicht es, den Prozessverlauf von den einzelnen Teilprozessen zu entkoppeln. Während die einzelnen Teilprozesse in EPK abgebildet werden, wird der Prozessverlauf in Ereignisdiagrammen modelliert. Dabei wird folgenden Prinzipien gefolgt:

- In einem Ereignisdiagramm wird durch eine gerichtete Verbindung von Ereignis e1 zu Ereignis e2 ausgedrückt, dass aus dem Bereitstellungsereignis oder dem externen Auslöseereignis e1 das interne Auslöseereignis e2 folgt.

- Bei den Startereignissen einer EPK handelt es sich ausschließlich um (interne oder externe) Auslöseereignisse.

- Ist ein von einer Funktion ausgelöstes Ereignis im Rahmen des Prozessverlaufs unweigerlich auch das auslösende Ereignis der nachfolgenden Funktion, so repräsentiert dieses sowohl ein Bereitstellungs- als auch ein internes Auslöseereignis, und wird nicht in einem Ereignisdiagramm aufgeführt.

---

[5] Diese Unterscheidung wird auch in [BKR05] diskutiert.

Das für die Idee der selbststeuernden Prozesskonfiguration wesentliche Kopplungssystem wird demzufolge vollständig in Ereignisdiagrammen abgebildet. Die daraus resultierende lose Kopplung der EPK führt zu einer weitgehenden Modulautonomie und infolgedessen Modul-Interflexibilität, da die Bedingungen bezüglich der Modulreihenfolge nicht in die EPK integriert sind.

Die Unterscheidung von Auslöse- und Bereitstellungsereignissen führt in Kombination mit den explizit formulierten Prinzipien zur Erstellung von Ereignisdiagrammen nicht nur zu Modul-Interflexibilität, sondern bildet auch die Basis für ein Vorgehensmodell zur Erstellung flexibler Prozessmodelle.

Ein Gesamtprozess besteht aus mehreren Teilprozessen (Modulen). Zwei Teilprozesse p1 und p2 sind durch ein gemeinsames Ereignis e1 verbunden. Das Ereignis e1 wird in der Folge nur noch als Bereitstellungsereignis, nicht mehr jedoch als Auslöseereignis klassifiziert. Es wird ein neues Auslöseereignis e2 eingeführt, welches e1 als Startereignis des Teilprozesses p2 ablöst (e1 bleibt unverändert Endereignis von p1). Die Teilprozesse p1 und p2 sind nun durch die explizite Formulierung von Auslöse- und Bereitstellungsereignissen entkoppelt. Es wird ein Ereignisdiagramm angelegt, welches die Transformation des Bereitstellungsereignisses e1 in das Auslöseereignis e2 explizit beschreibt. Dabei können weitere (externe) Auslöseereignisse, welche Umwelteinflüsse beschreiben oder Prozess(ablauf)konfigurationen spezifizieren, in die Transformationsregeln einbezogen werden.

Ein wesentliches Merkmal dieses Vorgehensmodells besteht darin, dass zu Beginn der Modellierung eines Prozesses nicht sämtliche „denkbaren“ Prozessverläufe abgebildet werden müssen. Die hier skizzierte Methodik führt durch die Einführung und kontinuierliche Erweiterung eines Kopplungssystems zu autonomen Teilprozessen und damit zu Modul-Interflexibilität.

Der hier vorgestellte Ansatz beschreibt die Module in EPK. Diese Module werden durch das Kopplungssystem, welches in Ereignisdiagrammen modelliert wird, situationsabhängig miteinander verknüpft.

Die in den Ereignisdiagrammen vorgenommene Abbildung von Bereitstellungs- und externen Auslöseereignissen auf interne Auslöseereignisse kann formal im Sinne wissensbasierter Regelsysteme mit Mitteln der mathematischen Logik beschrieben werden. Damit können in diesem Sinne verwendete Ereignisdiagramme als Regelsysteme interpretiert werden, auf die das Prinzip der Vorwärtsverkettung[6] angewendet wird.

Die Komposition des Gesamtprozesses aus den Teilprozessen (EPK) und dem Kopplungssystem (Ereignisdiagramme) kann durch folgenden Algorithmus skizziert werden:

---

[6] Vgl. zur Vorwärtsverkettung [Bi93] und [GFH90].

1. Bestimmung der Menge aller externen Auslöseereignisse.

2. Bestimmung der Menge aller internen Auslöseereignisse mittels Vorwärtsverkettung auf Basis der Ereignisdiagramme und der vorliegenden Bereitstellungs- und Auslöseereignisse.

3. Bestimmung aller EPK, deren Menge der Startereignisse eine Teilmenge[7] der Menge aller Auslöseereignisse ist (unter Berücksichtigung der durch Konnektoren in den EPK beschriebenen Verknüpfungslogik).

4. Durchlauf aller in Schritt 3 ermittelten EPK. Nach jedem Durchlauf einer EPK wird Schritt 2 wiederholt.

Der Algorithmus terminiert, wenn alle EPK durchlaufen sind und keine weitere EPK durch die Menge der vorliegenden Auslöseereignisse instanziiert werden kann.

Die strikte Trennung der EPK und Ereignisdiagramme nach dem Prinzip der selbststeuernden Prozesskonfiguration bietet zukunftsgerichtet einen weiteren Vorteil: sie ermöglicht die anwendungsfallbezogene[8] Komposition eines Prozessmodells. Während bei der klassischen Darstellung aus bestehenden Modellen Teile entfernt werden müssen, ermöglicht es das modul-interflexible Kopplungssystem, den Gesamtprozess aus den EPK der Teilprozesse (wie hier beschrieben) zusammenzusetzen; nicht benötigte EPK können einfach weggelassen werden, ohne dass sie modifiziert werden müssen. Damit wird die Basis für die Komposition von Prozessmodellen unter Berücksichtigung konkreter Anwendungsfälle geschaffen.[9]

Das Prinzip der selbststeuernden Prozesskonfiguration soll nun an einem einfachen Beispiel erläutert werden:

---

[7] Sind Startereignisse durch XOR-Konnektoren verknüpft, darf von diesen Startereignissen nur genau eines in der Menge der Auslöseereignisse enthalten sein.
[8] Zu Anwendungsfällen vgl. [Co01].
[9] Wenngleich eine EPK aufgrund ihrer Modul-Intraflexibilität eine Menge ähnlicher Anwendungsfälle abdecken kann.

Abbildung 1: Starrer Prozess zur Auftragsbearbeitung

Nach der Modellierung des in Abbildung 1 abgebildeten Prozesses wird festgestellt, dass auftragsabhängig eine Freigabe des Auftrags vor der eigentlichen Auftragsbearbeitung durchzuführen ist. Dies bedeutet, dass das externe Ereignis „Auftrag ist erteilt“ nicht mehr unmittelbar das auslösende Ereignis für den Teilprozess zur Auftragsbearbeitung darstellt. Der gemäß dem Prinzip der selbststeuernden Prozesskonfiguration angepasste Prozess lässt sich durch folgende Modelle abbilden:

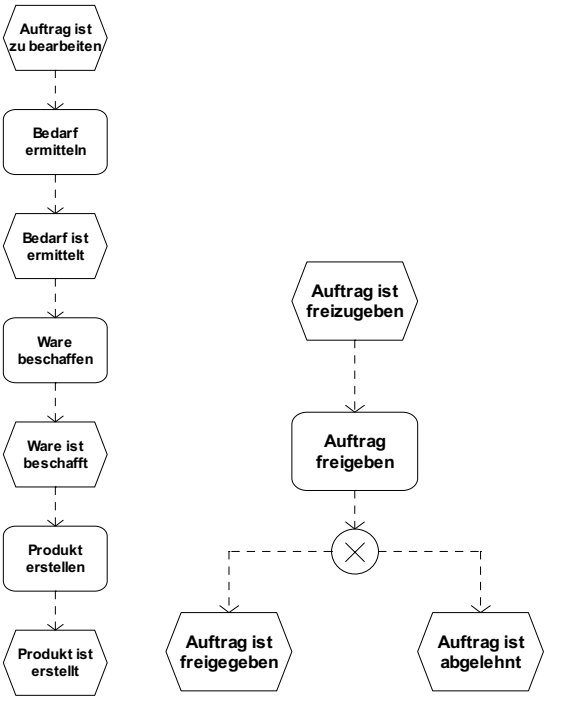

Abbildung 2: Flexibler Prozess zur Auftragsbearbeitung (EPK)

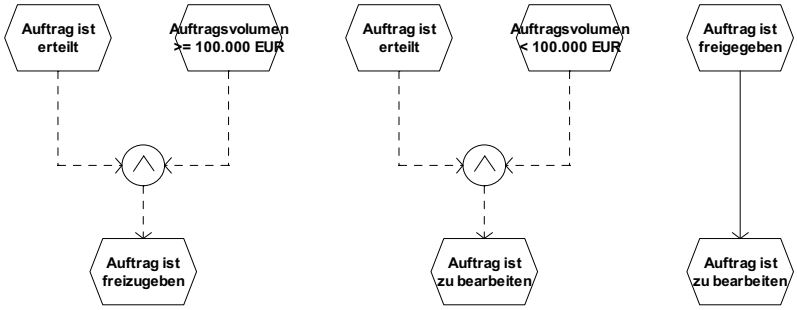

Abbildung 3: Flexibler Prozess zur Auftragsbearbeitung (Ereignisdiagramm)

In der ursprünglichen EPK zur Auftragsbearbeitung wurde lediglich das externe Auslöseereignis „Auftrag ist erteilt“ durch das interne Auslöseereignis „Auftrag ist zu bearbeiten“ substituiert. Ansonsten bleibt diese EPK unverändert. Sie ist völlig unabhängig von der EPK mit der Funktion „Auftrag freigeben“. Jede der beiden EPK enthält somit losgelöst vom Gesamtprozess eine isolierte Darstellung des betreffenden Teilprozesses. Das in dem Ereignisdiagramm abgebildete Kopplungssystem beschreibt die eigentliche Prozesslogik. Hier kann der Gesamtprozess flexibel an sich ändernde Anforderungen bezüglich der Notwendigkeit einer Auftragsfreigabe angepasst werden. Das Ereignisdiagramm beinhaltet damit letztlich die Prozesskonfiguration, welche für die Komposition der autonomen Teilprozesse zum Gesamtprozess verantwortlich ist.

Zum Vergleich seien die beiden klassischen Möglichkeiten in den Abbildungen 4 und 5 aufgezeigt, den Beispielprozess ohne das Konzept der selbststeuernden Prozesskonfiguration darzustellen.

Der entscheidende Nachteil in Abbildung 4 liegt in der engen Kopplung aller möglichen Prozessverläufe sowie der integrierten Abbildung von fachlichen Funktionen und Prozesslogik. Auftragsbearbeitung und –freigabe sind hier nicht als autonome Teilprozesse dargestellt, sondern gemäß der aktuellen Prozesslogik miteinander verknüpft. Sie sind infolgedessen keine autonomen Elemente einer Bibliothek von Teilprozessen.

Ungeachtet dessen, dass der Ablauf einer Freigabe für Aufträge >= 100.000 EUR in Bezug auf Aufträge mit geringen Beträgen ohne Belang ist, trägt er zur Komplexität der Darstellung des Ablaufs für Aufträge < 100.000 EUR bei. Die integrierte Darstellung der Prozessverläufe hat zur Folge, dass jede Änderung der Prozesslogik (beispielsweise zusätzlich die Berücksichtigung „priorisierter“ Aufträge) eine Anpassung der vorliegenden EPK bedingt, obwohl die Funktionalität der eigentlichen Auftragsbearbeitung davon unberührt bleibt.

Die hier verdeutlichten Probleme werden durch die Einführung von Prozessschnittstellen und die Verteilung der Abläufe auf mehrere EPK nicht grundlegend behoben, wie Abbildung 5 zeigt:

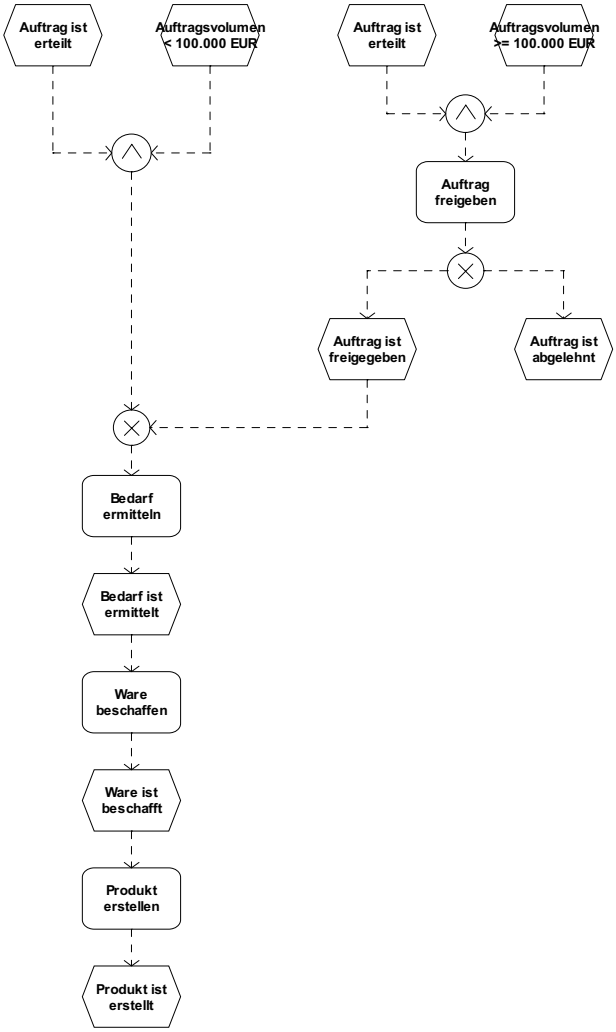

Abbildung 4: Unflexibler Prozess zur Auftragsbearbeitung (eine EPK)

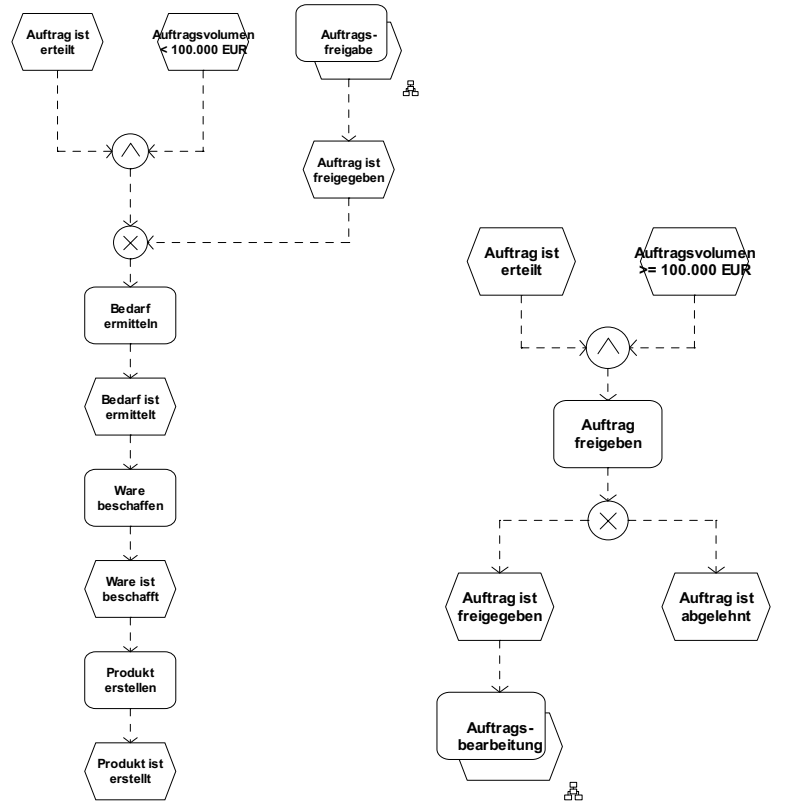

Abbildung 5: Unflexibler Prozess zur Auftragsbearbeitung (zwei EPK)

Durch die Einführung der Prozessschnittstellen wird zwar die interne Logik der Freigabe von der eigentlichen Auftragsbearbeitung getrennt, die EPK zur Auftragsbearbeitung bleibt jedoch weiterhin fest mit der (EPK zur) Freigabe verknüpft. Allgemein erfordert in der klassischen Darstellung eine Änderung der Modulreihenfolge die Anpassung der beteiligten EPK. Die Autonomie der Teilprozesse (EPK) ist durch die Prozessschnittstellen und die gleichzeitige Verwendung der Startereignisse als Endereignisse (und umgekehrt) verletzt.

Die selbststeuernde Prozesskonfiguration sorgt durch die Entkopplung der einzelnen EPK und die Kapselung des Prozessverlaufs in Ereignisdiagrammen für eine erhöhte Modul-Interflexibilität. Die Module können in nahezu beliebiger Reihenfolge durchlaufen werden.

Die aus der selbststeuernden Prozesskonfiguration resultierende höhere Modellzahl sowie die Einführung zusätzlicher Ereignisse in Form von internen Auslöseereignissen sind nur dann nicht zu rechtfertigen, wenn bei gering komplexen Problemstellungen wenige, geringfügige Unterschiede im Prozessverlauf abzubilden sind.

Es sei darauf hingewiesen, dass eine Substitution der Ereignisdiagramme durch entsprechende EPK, in denen die Bereitstellungs- und Auslöseereignisse durch Verknüpfungsfunktionen miteinander verbunden werden, möglich ist. Die Verwendung eines eigenen Modelltyps (Ereignisdiagramm) für die Verknüpfung der Module führt jedoch zu einer weitaus höheren Transparenz: die Module werden durch EPK, der Prozessverlauf wird durch Ereignisdiagramme modelliert. Zudem wird die Verknüpfungslogik nur in den Ereignisdiagrammen explizit abgebildet: sie können ohne zusätzliches Wissen über die Semantik der Verknüpfungsfunktionen interpretiert und somit maschinell ausgewertet werden.

4.2 Variable Funktionsreihenfolgen

Die Forderung nach Modul-Intraflexibilität lässt sich verschiedenartig konkretisieren. So können Umwelteinflüsse es erforderlich machen

- Funktionen bzw. Prozessschritte zusammenzufassen,
- einzelne Funktionen bzw. Prozessschritte nicht auszuführen,
- Funktionen bzw. Prozessschritte in unterschiedlicher Reihenfolge durchzuführen,
- auf Fehlersituationen unterschiedlich zu reagieren,
- die möglichen Ergebnisse einer Funktion bzw. eines Prozessschrittes situationsabhängig einzuschränken oder
- Funktionen bzw. Prozessschritte unterschiedlich zu parametrisieren

In diesem Abschnitt wollen wir uns der Thematik variabler Funktionsreihenfolgen zuwenden, welche in ihren einzelnen Ausprägungen schon Gegenstand zahlreicher Arbeiten gewesen ist.

Kiepuszewski behandelt in [Ki02] allgemeine Aspekte des „Interleaved Parallel Routing“. Gegenstand dieser Thematik ist die sequentielle Ausführung von Funktionen in einer Reihenfolge, die erst zum Zeitpunkt der Prozessausführung determiniert wird. Unter besonderer (aber nicht ausschließlicher) Berücksichtigung der Umsetzung in Petri-Netzen wird dieselbe Thematik ebenfalls in [Aa02] behandelt. In [MNN05] wird ein Ansatz zur Realisierung dieses Modellierungsmusters mit Hilfe von EPK vorgestellt.

Einfacher und in ihrer Ausrichtung ähnlich sind die Konzepte in [ST05] und [Gr05]. Während Scheer und Thomas einen neuen Operator einführen, den Sequenzoperator, kommt Grief in seiner Arbeit ohne zusätzliches Konstrukt aus. Er verwendet eine allgemeine Funktion (welche situationsbedingt spezialisiert werden kann) „nächste Aktion bestimmen“ in Kombination mit einem XOR-Verteiler und einer Schleife. Durch den XOR-Verteiler wird explizit gemacht, dass die zur Auswahl stehenden Prozesspfade alternativ durchlaufen werden, und eine Parallelität aus Prozesssicht nicht erlaubt ist. Durch die Schleife wird die Entscheidungsfunktion „Nächste Aktion bestimmen“ (oder eine prozess- bzw. situationsabhängig spezialisierte Funktion) wiederholt durchlaufen, wodurch eine dynamische Entscheidung bezüglich des nächsten auszuführenden Prozesspfades zur Laufzeit des Prozesses ermöglicht wird. Dabei kann auch entschieden werden, dass situationsbedingt nicht alle Prozesspfade durchlaufen werden. Ein konkretes Beispiel zur Anwendung dieses Modellierungsmusters ist in [Gr05, S. 202 - 205] zu finden.

5 Gesamtkonstruktion

Während mit der selbststeuernden Prozesskonfiguration Modul-Interflexibilität erreicht wird, besteht die Zielsetzung variabler Funktionsreihenfolgen in der Modul-Intraflexibilität. Für eine Zusammenführung der beiden Ansätze ist die Wahl der Betrachtungsebene, „intermodular“ oder „intramodular“ zu klären: Kann also eine vorliegende Flexibilitätsproblematik prozessbezogen durch ein Modul oder nur durch mehrere Module abgebildet werden.

Die selbststeuernde Prozesskonfiguration mit Hilfe von Ereignisdiagrammen setzt voraus, dass die Umwelteinflüsse, welche die nächste auszuführende Funktion bzw. den nächsten Prozessschritt determinieren, explizit durch Ereignisse benannt und abgebildet werden können. Die in [ST05] und [Gr05] beschriebene variable Funktionsreihenfolge zur Erreichung von Modul-Intraflexibilität kann auch dann bereits angewendet werden, wenn die alternativen Prozesspfade bekannt sind, jedoch noch bezüglich der Bedingungen, unter denen der nächste auszuführende Prozesspfad auszuführen ist, Unsicherheit herrscht.

Der Einsatz der selbststeuernden Prozesskonfiguration bietet sich „Top-down“ folglich an, wenn in einer Flexibilität fordernden dynamischen Umwelt die Gesamtsituation durch Ereignisdiagramme höher aggregiert beschreibbar ist. Einzelne, nicht vollständig durch Ereignisse erfassbare Unsicherheiten der Gesamtsituation können durch variable Funktionsreihenfolgen in Modulen detailliert abgebildet werden.

„Bottom-up“, von einem flexiblen Modul (zur Bewältigung einer Teilproblematik) zu einer flexiblen Modulmenge (für die Gesamtproblematik), kann gegangen werden, wenn sich zunehmend Unsicherheiten durch klar definierte Ereignisse darstellen lassen.

Grundsätzlich denkbar ist aber auch, beide Ansätze (selbststeuernde Prozesskonfiguration und variable Funktionsreihenfolgen) auf einer Betrachtungsebene anzuwenden. Dies sei anhand des bereits bekannten Beispiels zur Auftragsbearbeitung veranschaulicht:

Zu Beginn der Prozessmodellierung ist bekannt, dass unter – noch nicht geklärten Voraussetzungen – ein Auftrag zunächst freizugeben ist, bevor er abgewickelt werden darf. Damit sind die Voraussetzungen zur Abbildung der Prozesskonfiguration in einem Ereignisdiagramm noch nicht gegeben. Die in [ST05] und [Gr05] beschriebene Methode zur Abbildung variabler Funktionsreihenfolgen kann jedoch bereits angewandt werden. Herrscht später Klarheit über die konkreten Bedingungen für die Notwendigkeit einer Freigabe, so sind die Voraussetzungen für eine selbststeuernde Prozesskonfiguration erfüllt.

Literaturverzeichnis

[Aa02] van der Aalst, W.M.P.: Workflow Patterns, QUT Technical Report. Queensland University of Technology, Brisbane, 2002.

[Al98] Allweyer, T.: Adaptive Geschäftsprozesse. Gabler, Wiesbaden, 1998.

[AS95] Allweyer, T, Scheer, A.-W.: Modellierung und Gestaltung adaptiver Geschäftsprozesse. Institut für Wirtschaftsinformatik der Universität des Saarlandes, Heft 115, 1995.

[AS04] Aier, S., Schönherr, M.: Enterprise Application Integration als Enabler flexibler Unternehmensarchitekturen. In (Hasselbring, W., Reichert M., Hrsg.): EAI 2004 – Enterprise Application Integration. Tagungsband des GI-/GMDS-Workshops EAI'04, OFFIS, Oldenburg, 12. – 13.Februar 2004.

[Ba91] Barney, J.: Firm Resources and Sustained Competitive Advantage. In: Journal of Management, 17 (1991) Nr. 1, S. 99 – 120.

[Bi93] Bibel, W.: Wissensrepräsentation und Inferenz. Vieweg, Wiesbaden, 1993.

[BKR05] Becker, J., Kugeler, M., Rosemann, M. (Hrsg.): Prozessmanagement. 5. Auflage. Springer, Berlin, Heidelberg, New York, 2005.

[BMH05] Beimborn, D., Martin, S. F., Homann, U.: Capability-oriented Modeling of the Firm. In: Proceedings of the IPSI 2005 Conference; Amalfi/Italien (ohne Seitenangaben).

[Co01] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, 2001

[CS94] Chen, R., Scheer, A.-W.: Modellierung von Prozeßketten mittels Petri-Netz-Theorie. Institut für Wirtschaftsinformatik der Universität des Saarlandes, Heft 107, 1994.

[Da99] Dandl, J.: Objektorientierte Prozeßmodellierung mit der UML und EPK (Arbeitspapiere WI Nr. 12/1999, Lehrstuhl für Allg. BWL und Wirtschaftsinformatik, Univ.-Prof. Dr. Herbert Kargl). Universität Mainz, 1999.

[Da03] Davis, R.: Business Process Modelling with ARIS. Springer, 2003.

[Ec2005] Economist Intelligence Unit (Hrsg.): Business 2010 – Embracing the challenge of change. http://www.eiu.com/site_info.asp?info_name=eiu_SAP_business2010 (28.09.2005).

[Fe86] Feiland, F.-M.: Strategien erfolgreicher mittelständischer Unternehmen (Schriften zur Mittelstandsforschung Nr. 42-1986). C.E. Poeschel, Stuttgart, 1986.

[GFH90] Gottlob, G, Frühwirth, T, Horn, W.: Expertensysteme. Springer, Wien, New York, 1990.

[Gr05] Grief, J.: ARIS in IT-Projekten. Vieweg, Wiesbaden, 2005.

[GSW98] Glaser, H., Schröder, E. F., Werder, A. v. (Hrsg.): Organisationen im Wandel der Märkte. Gabler, Wiesbaden, 1998.

[HC94] Hammer, M., Champy, J.: Business Reengineering. Campus, Frankfurt/M., New York. 1994.

[Jo05] Jost, W.: „Der Ball muss ins Tor". In: SCHEER Magazin, 14 (2005) Nr. 3, S. 6 – 9.

[Ki02] Kiepuszewski, B.: Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows, PhD thesis, Queensland University of Technology, Brisbane, Australia, 2002.

[KK92] Kieser, A., Kubicek, H.: Organisation. 3. Auflage. de Gruyter, Berlin, 1992.

[KNS92] Keller, G., Nüttgens, M, Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)". Institut für Wirtschaftsinformatik der Universität des Saarlandes, Heft 89, 1992.

[LA98] Loos, P., Allweyer, T.: Process Orientation and Object-Orientation – An Approach for Integrating UML and Event-Driven Process Chains (EPC) (Publication of the Institut für Wirtschaftsinformatik, University of Saarland, Saarbrücken, Paper 144). Saarbrücken, 1998.

[MNN05] Mendling, J.; Neumann, G.; Nüttgens, M.: Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). Second GI-Workshop XML4BPM, Karlsruhe 2005.

[Oe2003] Oestereich, B., Weiss, C., Schröder, C., Weilkiens, T., Lenhard, A.: Objektorientierte Geschäftsprozessmodellierung mit der UML. Dpunkt.verlag, Heidelberg, 2003.

[PDF05] Picot, A., Dietl, H., Franck, E.: Organisation. 4. Auflage. Schäffer-Poeschel, Stuttgart, 2005.

[Ru99] Rump, F., J.: Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten. Teubner, 1999.

[Sc01] Scheer, A.-W.: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen, 4. Auflage, Springer, Berlin usw., 2001.

[Sc03] Schreyögg, G.: Organisation. 4. Auflage. Gabler, Wiesbaden, 2003.

[Se05] Seidlmeier, H.: Informationssysteme und Unternehmensprozesse als wettbewerbskritische Ressourcenbündel (in Vorbereitung).

[SK03] Siedersleben, J., Kurpjuweit, S.: Systemübergreifende Software-Architektur: Erfahrungen und Thesen. In: (Sinz, E. J., Plaha, M., Neckel, P. Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003. Proceedings der Tagung MobIS, Bamberg, 9. – 10. Oktober 2003.

[SNZ97] Scheer, A.-W., Nüttgens, M., Zimmermann, V.: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) – Methode und Anwendung (Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes, Heft 141). Saarbrücken, 1997.

[ST05] Scheer, A.-W., Thomas, O.: Geschäftsprozessmodellierung mit der Ereignisgesteuerten Prozesskette. In: Das Wirtschaftsstudium 34 (2005) Nr. 8-9, S. 1069-1078.

[Ut97] Uthmann, C.: Nutzenpotenziale der Petrinetztheorie für die Erweiterung der Anwendbarkeit Ereignisgesteuerter Prozeßketten. Vortrag im Rahmen des Workshops an der Universität Oldenburg, 1997.

[Zi99] Zimmermann, V.: Objektorientiertes Geschäftsprozessmanagement. Deutscher Universitäts-Verlag, Wiesbaden, 1999.

My own process: Providing dedicated views on EPCs

Florian Gottschalk Michael Rosemann Wil M.P. van der Aalst

f.gottschalk@tm.tue.nl m.rosemann@qut.edu.au w.m.p.v.d.aalst@tm.tue.nl

Abstract: The idea of Business Process Management demands that companies change their focus from optimising tasks to focusing on whole business processes optimising the overall value chain. However process models depicting such complex processes are perceived as complicated and therefore as hard to use. The critical task is to present only relevant model parts to users and at the same time enable them to locate their contribution within the entire value chain.

This paper discusses an approach for tailoring Event-driven Process Chains to those parts that are relevant to selected organisational units. The approach uses the allocation of organisational units to functions as a selection criteria for relevant model parts. A distinction between concurrent and alternative collaboration and the implementation of a corresponding notation within the EPC notation enable the introduction of additional process interfaces, a standard feature of Event-driven Process Chains, into the tailored models. The process interfaces ensure the visibility of the connected business process. Therefore the approach helps to resolve the depicted conflict.

1 Introduction

In order to handle and accurately describe business processes for all parties involved, today's companies are using numerous modelling techniques, each aiming at different goals and audiences. This leads to a significant complexity of the modelling landscape. A high degree of complexity however results into a decrease of user acceptance [RSD05]. For that reason the quality of conceptual models is subject of academic research for a long time (e.g. [LSS94, Ros96]). When creating models, companies have to incorporate the same factors as for every other product, i.e. time, costs, and quality. The impact of these factors also depends on the purpose of modelling. E.g., a model created for simulation purposes will differ from the model created for knowledge management or organisational documentation. Also the HR manager's demand on models of the company will for sure differ from the requirements of technicians. Proces modelling for various user groups or purposes is called *multi-perspective modelling* whereas each perspective is a subset of the total model [Ros03].

Powerful tools like the Architecture of Integrated Information Systems (ARIS) [Sch94b, Sch00] support the creation of such multi-perspective models. The architecture enables the integration of different perspectives. It distinguishes between an object and its oc-

currences. By using occurrences, the same object can be used in several models and modelling perspectives. Within ARIS, the process modelling language of *Event-driven Process Chains* (EPCs) is used as an anchor point for the integration of the different perspectives. EPCs are commonly used to depict the control flow of a business process, i.e. the order in which tasks have to be performed (see Figure 1). In addition EPCs provide the integration of multiple perspectives. They allow connecting occurrences of elements used within specialised perspectives (e.g. data, organisational units, or utilities) to functions. So the relevance of the particular element for the function becomes obvious. However, the current assignment notation lacks of information about the interaction between multiple connected elements. For that reason we introduce new connectors which are depicting different kinds of collaboration in the first part of this paper (e.g., see function *Release Invoice manually* in Figure 1). We focus on the assignment of organisational units to functions, but connectors for other assignments should be definable in a similar way.

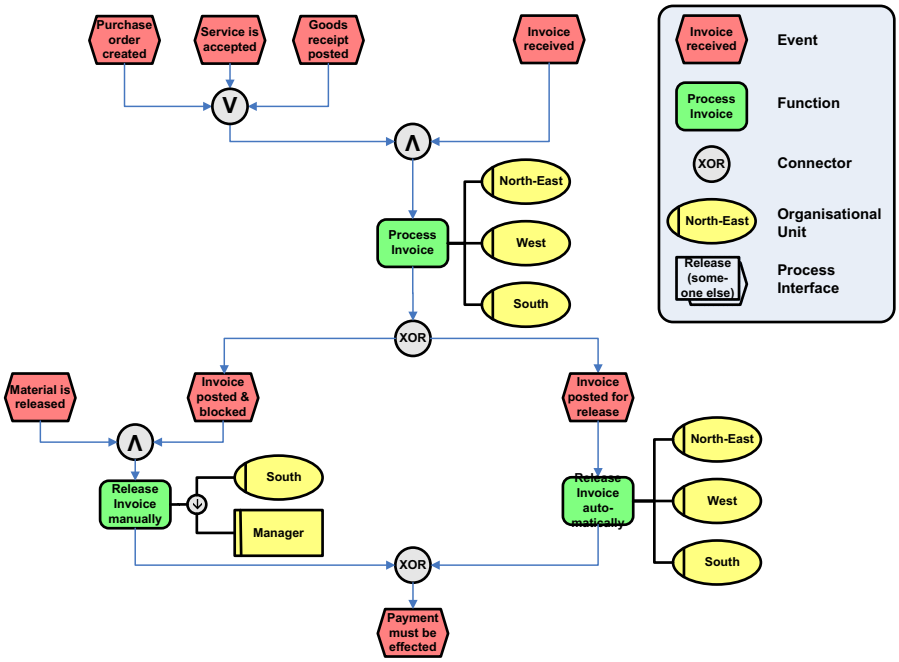

Figure 1: Example Invoice Verification Process

By adding additional elements like stakeholders or data to the process flow even small EPCs can become fairly complex [KKS04]. E.g., the simple and clearly arranged invoice verification process in Figure 1 needs almost half an A4 page. Also the structure of EPCs (with, functions, surrounding events, connectors, and arcs in between) drives the model complexity, especially when creating models with a large number of functions and connectors. However, it is important that a user of a model quickly identifies those parts of the

model that are relevant to him, i.e. the level of information presented must correspond to his requirements [MG75, BDFK03]. Thus, we introduce a reduction mechanism for EPCs in the second part of the paper so that the resulting process model is of high relevance for a selected organisational unit. E.g., in the depicted invoice verification process the manager should just see the function *Release Invoice automatically* and its direct environment. For this we not only consider the selected elements from the original process (similar to [BDFK03, BDKK02, RSD05]) but also introduce interfaces making the overall process flow and therefore the contribution to the value chain visible.

To conclude we summarise the contribution of this paper and we give an outlook on potential future extensions.

2 Assigning Organisational Units to Functions: Who has to do it?

To depict the involvement of organisational units within a process, EPCs allow connecting organisational units to functions. The reasons for such a connection (and therefore for the involvement) are manifold. E.g., the ARIS Toolset [Sch94a] suggests reasons like:

- The organisational unit executes the function.
- The organisational unit contributes to the function.
- The organisational unit must be informed about result of the function.
- The organisational unit has a consulting role in the function.

In the definition of eEPCs it is requested that each organisational unit is involved in at least one function and that each function is allocated to at least one organisational unit as depicted in the meta model in Figure 2.

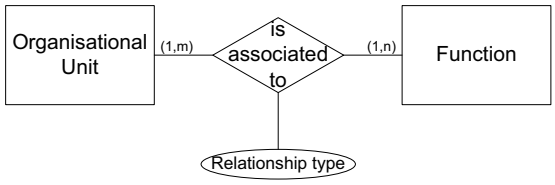

Figure 2: Meta model for allocating organisational units and functions (adapted from [RzM97, Sch00])

The meta model depicts that an organisational unit can be involved in more than one function as well as more than one organisational unit can be involved in a function – also for the same reason. E.g., it may be required that the management as well as the sales department have to contribute actively to the performance of a market analysis. Table 1 provides a matrix illustration of the involvement of different departments in a real-world order processing situation where three kinds of involvement are distinguished [Sch00]. The notion of the matrix quasi conforms to so-called RACI matrices [PW05], just the labeling differs.

Organizational Units / Functions	Management	Marketing	Management & Organization	Branch Management	HR	Cost Acc. & Controlling	Sales	Sales and Distribution	R&D	Production	Purchasing / Procurement	Materials Warehouse
Market Analysis	i	r	a	a			i					
Production Program Planing	r	i	i	a	a	a	i		a	a	a	
Proposal Processing							r					
Order Processing							r					
Product Development	i			a	a	i	i		r	i	i	
Production Planing			i		i	a	i		i	r	i	a
Materials Purchasing											r	i
Warehouse Management								a				r
Prroduction Mgt. & Control						a	a					
Quality Assurance				a			a	a	i	r	i	r
Shipping				i			i	r				
Cost Accounting & Control	a		a	i	a	r			a	a	a	
Financial & Investment Plan.	i		r	a	a	i				a	a	
HR Planning & Development	i		i	a	r	a				a	a	
Inventory & Year-end Closing	i		a	i		r				a	a	

r = responsible i= actively involved a=associated

Table 1: Matrix illustration of a function allocation [Sch00]

Neither the matrix nor a corresponding EPC depicts interrelations between different organisational units involved in the same way to the same function. E.g., it might be possible that not both management and sales are required but rather each of them is able to perform a market analysis. That means several organisational units can be involved in a function concurrently or alternatively. Obviously this makes not only a huge difference for ressource utilisation, but also influences the synchronisation of the individual user tasks. So the opportunity to specify this accurately should be provided by the modelling environment.[1]

[1]The simulation of the ARIS Toolset offers the opportunity to specify if allocated organisational units are involved either concurrently or alternatively by maintaining Boolean attributes. It is not possible to specify a combination of concurrent and alternative involvement in the same function nor to specify different types of resource allocation for different reasons. Both might be required if more than two organisational units can be or are involved in a function.

Hence we suggest to use a logical term for specifying such relationships. E.g., the active involvement for market analysis in Table 1 can be depicted as *Management* AND *Sales* for concurrent involvement or as *Management* XOR *Sales* for alternative involvement. In product development more than two organisational units are actively involved, e.g., the relationship might be *Purchasing / Procurement* AND *Cost Acc. & Controlling* AND (*Management* XOR *Production* AND *Sales*). Within this notation it is assumed that precedence is the same as in Boolean algebra or common programming languages, i.e. the precedence of the AND operator is higher than the precedence of the XOR operator. The notation also allows depicting relationships like "two out of three associated organisational units are required". E.g., if *A*, *B*, *C* are the possible organisational units the term would be *A* AND *B* XOR *A* AND *C* XOR *B* AND *C*.

To depict such relationships within an EPC process model, all terms must be expanded until brackets are not required anymore. E.g., for the active involvement in product development that would result in the association *Purchasing / Procurement* AND *Cost Acc. & Controlling* AND *Management* XOR *Purchasing / Procurement* AND *Cost Acc. & Controlling* AND *Production* AND *Sales*. This distinguishes clearly both possible involvement combinations. It allows grouping all organisational units that are required within each combination by an AND-connector. Afterwards the AND-connectors can be connected to the function. Then all connections of organisational unit groups to the function are the alternatively involved organisational unit groups. The resulting model for this example is displayed in Figure 3.

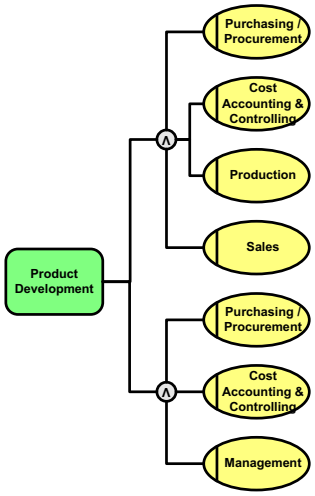

Figure 3: Multiple Organisational Units associated to a function in two alternative groups

In addition to the distinction between alternative and concurrent involvement, the concurent involvement can be further specified. E.g., it makes a difference for involved organisational units, if the task is a meeting where all organisational units have to perform the

task together at the same time, or if each organisational unit gives its own contribution to a function that can be executed in parallel to and independent of other organisational units. A third option would be that the organisational units are involved in a certain order. In general, it is possible to distinguish between *joined*, *parallel*, and *sequential* involvement. To make this distinction visible, the AND-connector may be converted to specialised connectors depicting the particular relationship. E.g., the joined involvement can be depicted by the AND-symbol, the parallel involvement by two vertical lines, and the sequential involvement by an arrow showing the order of involvement as depicted in Figure 4.

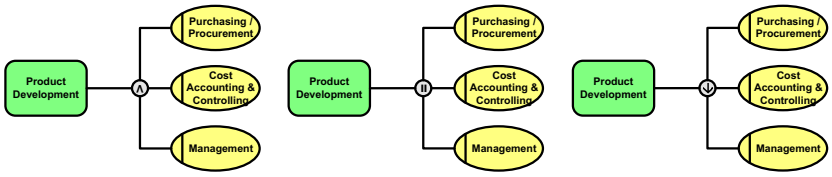

Figure 4: Possible illustration of joined, parallel, and sequential involvement

Taking Guidelines of Modelling [Ros96, BRS95] into consideration, such new, specialised symbols can be put into question. E.g., the collaboration could also be specified through the process flow in a hierarchical lower level EPC. In fact, it depends on the purpose of modelling if the relationship should be depicted within the model, maintained in attributes, and/or specified on a lower level EPC. When making this decision the matching of levels between the organisational hierarchy and the process model hierarchy must be taken into consideration as well.

To depict grouped association between organisational units and functions within the meta model the entity type *organisational unit grouping* must be introduced. The entity type can represent either a joined, a parallel, or a sequential collaboration. In case of a sequential collaboration also the position of the organisational unit within the sequence must be maintained. This is done in the association between the *organisational unit* and the *organisational unit group* (see Figure 5).

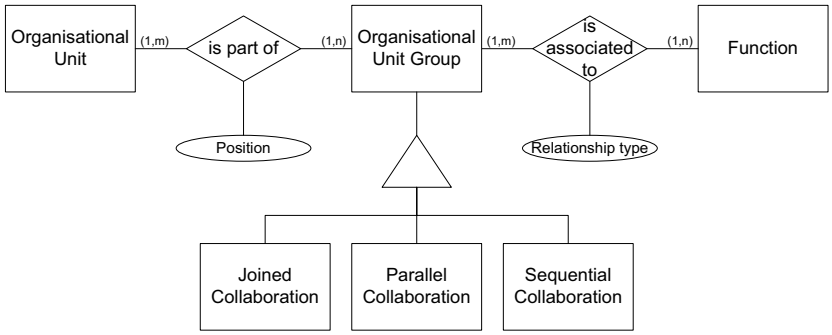

Figure 5: Meta model for allocating organisational units to functions via organisational unit groups

3 Deriving Individual Process Models: What do I have to do?

Handling of complexity and adjusting the scope of models to the requirements of the user is of prime importance to facilitate the usage of models ("Guideline of relevance" [Ros96, BRS95]). The allocation of organisational units to functions can be used as a selection criteria. It enables hiding of unselected process elements which are irrelevant to the particular organisational unit. I.e. the complex model is reduced to clearer, smaller models and the model's degree of relevance increases (see [BDFK03]). However, in this case the user is not aware of the overall business process as all visibility of process elements in which the particular organisational unit is not involved in is removed. Such an approach would conflict with the idea of Business Process Management that demands comprehensive business processes addressing the whole value chain and not ending at the borders of an organisational unit [HC93].

Indeed we argue in our approach that functions in which the considered organisational unit is not involved are irrelevant and remove them from the models. Nonetheless it is important to depict the link to these functions so that the business process flow is kept visible. Consequently, our approach uses the concept of process interfaces within EPCs. Process interfaces are navigation aids that show the link from one to another process [KT98]. They visualise the connection between processes on the same hierarchical level [KKS04]. I.e. in our approach we introduce a process interface to the process model whenever other organisational units are involved in the same, preceding or succeeding task of a business process. So the process interfaces keep the inter-organisational unit process flow visible although irrelevant parts are not included in the specific model. E.g., Figure 6 depicts the invoice verification process from Figure 1 tailored for the manager. He is only confronted with

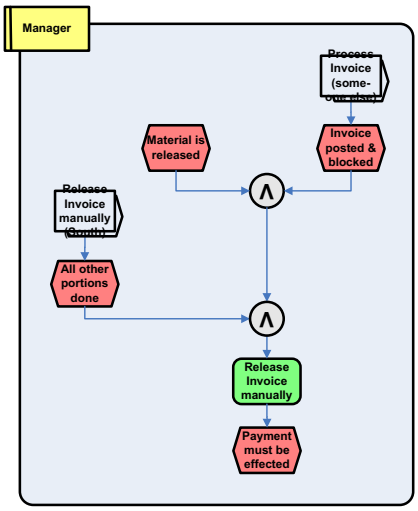

Figure 6: Manager's Invoice Verification Process

those parts of the process he is involved in and the process interfaces clearly depict what has to be done before he becomes involved. So the manager is aware of his contribution within the overall process flow but he is not confronted with irrelevant model parts.

The developed algorithm can be divided into five main steps:

1. Copy[2] all functions performable by the considered organisational unit into a new individual process model.
2. Copy for all these functions their preceding and subsequent (start- and end-) events as well as the in-between connectors and arcs from the original process model to the individual process model.
3. Analyse for each function all preceding and subsequent functions regarding their possible alternative executing organisational units. If required, introduce new process interfaces and events and integrate them into the individual process model.
4. Analyse for each function if it must be executed together or in parallel with other organisational units. If required, introduce new process interfaces and events and integrate them into the individual process model.
5. Remove unnecessary connectors from the individual process model.

The first step ensures that all functions that can be performed by the particular organisational unit appear in the individual process model as well. The performable functions can easily be derived from the EPC by analysing the connections of organisational units to functions. The second step forms an EPC around the copied functions by re-establishing its original environment. Thereby, and (if not otherwise depicted) in the following description of the algorithm the term *copy* is meant in the sense of "copy this occurrence and replace it if already existing". That means the same occurrence of an event also occurs in the individual model only once.

These two steps already derive a syntactically correct EPC that includes all executed functions and that conforms to the model derived by the element selection of [BDFK03]. However, the derived process models reflect only the parts of the process executed by the considered organisational unit. To keep the visibility of the overall business process, organisational breaks have to be analysed and additional entrance and exit points to and from the process have to be introduced as process interfaces. Such process interfaces will

[2] In fact, the algorithm does not hide non-required elements as suggested in [BDFK03]. Instead it copies occurrences of the required elements and adds the process interfaces to a new process model. This is done to address some further demands on tools for comprehensive process modelling. Among other requirements such a tool must allow maintaining and managing of all different modelling phases [BDKK02]. It might be required that a derived individual model for an organisational unit has to be modified, i.e. different organisational units end up in different process models [RA05, DCRvdA05]. If this would be done on the same model with hidden elements, each decision would effect the process within other organisational units. A high degree of standardisation and equal processes among different organisational units is indeed desirable, but it is not always practicable (e.g., because of different legal regulations in different countries). That means such a standardisation of individual processes for organisational units requires the consolidation of the individual process models back into a comprehensive business process model. However, this consolidation is out of the scope of this research.

depict that other organisational units are executing functions which are connected to the process of the particular organisational unit.

For this purpose it is required to analyse all preceding and all subsequent functions (below also called *connected functions* or *A*) for each of the copied functions (below also called *original function* or *B*) regarding the organisational units performing them (Figure 7). If a function is analysed by the algorithm, this reflects the runtime situation that the particular organisational unit performs this function. Thus it is irrelevant which other organisational units can perform the considered function itself alternatively. It is also sufficient to analyse the performance of directly preceding and succeeding functions. If these connected functions can be performed by the particular organisational unit (and therefore are relevant), they will be analysed by the algorithm on their own.

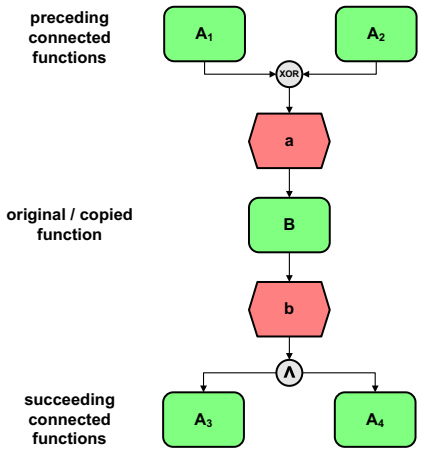

Figure 7: Preceding and succeeding functions

When regarding the groups of organisational units that are performing the connected functions, we have to distinguish between three types of groups:

- Only groups of organisational units to which the regarded organisational unit belongs can perform the function: That means the particular organisational unit has to perform this function in any case.
- Only groups of organisational units to which the organisational unit does not belong can perform the function: That means the particular organisational unit must not perform the function. As the regarded organisational unit is not involved in this connected function, it is irrelevant which other groups of organisational units are able to perform that function: The own process starts or stops at the event.
- Groups of organisational units to which the regarded organisational unit belongs as well as groups of organisational units to which the regarded organisational unit does not belong can perform the function: If in the particular instance of the process the

organisational unit is not involved in the connected function, it is irrelevant to the particular function which organisational unit group is involved as the own process starts or stops at the event.

Two functions within an EPC must always be connected via an event in between. In addition to the event, also one or more connectors can be located in between. However this has no influence on our approach and can therefore be neglected.

As some functions require not only a single organisational unit but whole groups of organisational unit for their execution, also the synchronisation of the individual processes for the involved organisational units must be ensured. This is done in the fourth algorithm step. As depicted in section 2 we have to distinguish if both organisational units must perform the functions together, in parallel or in a sequential order. Executing the function together means that all organisational units must perform the function at the same time, whereas executing the function in parallel means that the function is only completed if all involved organisational units have performed and finished their portion. Performing the function in a sequential order means that the required organisational units neither can perform the function at the same time nor they need to perform it together. They have to execute it one after another.

The final step is a cleanup-step. Some connectors copied by the algorithm will be connected to only two other elements. These connectors are not required and should be replaced with shortcuts.

The first two steps as well as the fifth step do not distinguish different scenarios depending on the context. However, the results of the third and fourth step depend on the context of the particular function. Therefore we will now analyse how the different situations can be handled. Afterwards we will provide a small example application.

3.1 Analysis of Scenarios for Alternative Execution

The analysis of Scenarios for alternative execution is grouped by the relation between the considered organisational unit and the organsiational units involved in the connected function:

Function must be executed by the same organisational unit

Obviously, if the preceding, connected function A must be executed by the same organisational unit as the regarded function B (see Figure 8), A has already been copied to the new process in the first algorithm step. Both functions ensure that the event in between is copied as well. Each function ensures that all connectors between the particular function and the event are copied. As the process flow between both functions does definitely not change the executing organisational unit no process interfaces to other organisational units have to be introduced. That means this situation does not require any further treatment in this algorithm step. Of course, the same argumentation will hold for the other way around,

i.e. if A would be the regarded function and B would be connected to it.

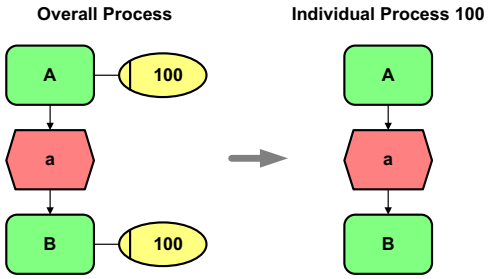

Figure 8: Functions executed by the same organisational unit

Function must be executed by one or more other organisational units

If a preceding function A has to be executed by one or more other organisational units this should be depicted in the individual process by replacing the function with a process interface. To insert this process interface it should be named like the function with the addition that it is performed by another organisation unit. To connect the process interface all arcs and connectors between the already copied event and the preceding function A (within the original process model) must be copied to the individual process – however the connection to A in the original model must be connected to the process interface instead. Connectors in between have no influence on the copy process. If connectors in between exist they are just copied into the resulting process as in Figure 9. However, only the arcs on the path between the function/process interface and the event are copied. If other arcs are required, they will be copied when regarding the particular connected function.

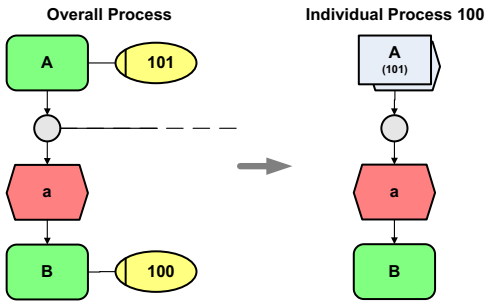

Figure 9: Preceding function, executed by another organisational unit

The process interface enables a clear identification of what has been done before the individual process starts as well as why it starts. It allows keeping track of the business

process. Also in this case the same argumentation would hold for a succeeding function. A separate description is therefore omitted.

Function can be executed by the same or other organisational units alternatively

If a function can be executed by the same as well as by other organisational units, we need to distinguish between preceding and succeeding connected functions. Of course, as above, if a preceding function A is executed by one or more other organisational units this should be depicted in the individual process as a process interface. However, in this scenario it is also possible that A is executed by the organisational unit itself (or a group of organisational units where it belongs to). For that reason the process interface must be introduced in addition to the function A which was already copied in step one of the algorithm. In step two the connection path between the function A and its end-event a was already copied to the individual process. To integrate the new process interface, the arc which is connecting A with the first succeeding connector or, if no connector exists, the arc which is connecting A with its end-event a must be removed. Instead of the arc function A and the new process interface must be connected to a new XOR-join-connector. Then the XOR-connector is connected to the element to which A was previously connected to (see Figure 10). As the arc that was connected directly to function A is broken up, connectors between A and event a have no further influence on the algorithm.

The introduced process interface depicts a new starting point of the process for the case that the regarded organisational unit does not contribute to function A. At the same time the opportunity is kept that the regarded organisational unit contributes to both functions.

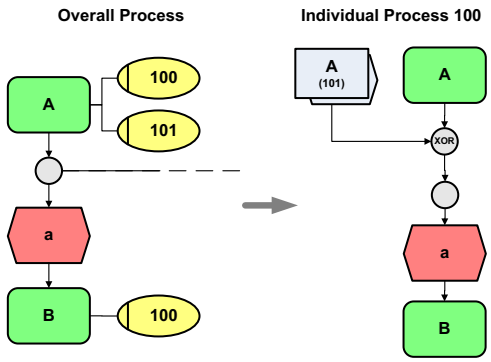

Figure 10: Preceding function, executed by the same or another organisational unit alternatively

The succeeding scenario is a bit more difficult to handle as one out of at least two organisational units (or groups) has to continue executing the process and a decision must be made who continues. Either the particular organisational unit can continue executing the process by performing the next function or the process is handed over to another organisational unit that is allowed to perform the succeeding function. Obviously the distinction between these two possibilities must be depicted as an XOR-split-connector in the individ-

ual process model. Theoretically, three options exist to introduce this connector. First of all, according to the cases above it could be introduced directly before function A. However, this would lead to an incorrect EPC as a split connector would follow an event and an event is not able to perform any decision. The second option would be to introduce the XOR-split-connector directly before event b. This would lead to a syntactically correct EPC. However, it implicates some semantic problems. Firstly, it adds additional functionality to the predefined function B. This could cause problems if a further specifying process model is assigned to the function that does not include the decision. Additionally, this solution will cause huge efforts if b is followed by an AND-split. According to the EPC rules this is permitted. However, introducing the XOR-split before b would require copying the AND-split as well. All decisions made for other connected functions to the end split (that are also analysed as they are connected to B as well) must then be transferred to the new branch in addition.

The third and preferred possibility is to introduce the decision task as a separate function that is followed by the XOR-split-connector with the end-events *proceed* and *other organisational unit proceeds*. The event *other organisational unit proceeds* is connected to the new process interface for depicting that in this case the process continues, but is not executed by this organisational unit. For resulting into a valid EPC the whole construct is inserted into the process directly before function A and behind any connector (see Figure 11). This solution does not add any additional functionality to a function. However, it involves a certain risk for overemphasising the decision task as it is depicted in the same way as other tasks. In addition, the function may lead to the assumption that a free choice for the employee will exist if he/she continues, whereas in practice probably a strict set of rules will exist. Thus the decision function is nothing else than applying this set of

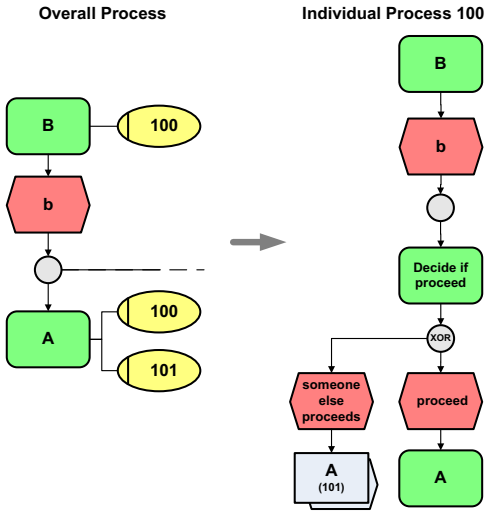

Figure 11: Succeeding function, executed by the same or another organisational unit alternatively

rules to the actual process. To handle pools of tasks, it is required to allow an implicit performance of the decision function. If more than one organisational unit including the regarded unit is allowed to execute function A, all of them will use the same in-tray that is at the same time the out-tray of function B. If function B was executed by the regarded organisational unit, and another organisational unit picks up the process to continue, the regarded organisational unit implicitly decides not to continue. To legitimate this form of decision making, e.g., it is possible to say the organisational unit made the decision by not continuing directly or by waiting too long before continuing.

3.2 Analysis of Scenarios for Concurrent Execution

After introducing new process interfaces for alternative executing organisational units, it must be analysed if additional organisational units are required to execute functions. Thus, each function executed by the regarded organisational unit must be analysed regarding additionally required organisational units. If additional organisational units are required, a synchronisation of the processes must be performed as follows.

Parallel performance

If a function must be performed by two or more organisational units in parallel – as depicted above that means that each organisational unit can start and perform its portion of work individually as soon as the function's start event has occurred – no organisational unit will have to wait for others. However, for completing the function and firing its end-event all organisational units involved must have finished their tasks. Thus, the synchronisation of the two (or more) processes should be performed directly after the function and therefore before all possible end-events. The synchronisation can be depicted as an AND-join-connector that fires the subsequent process flow only if all preceding paths are fired. It has to merge a new additional process interface with the organisational unit's process flow and should be named like the function executed in parallel added by a comment that this is the part of the function all others involved perform (see Figure 12). As the synchronising

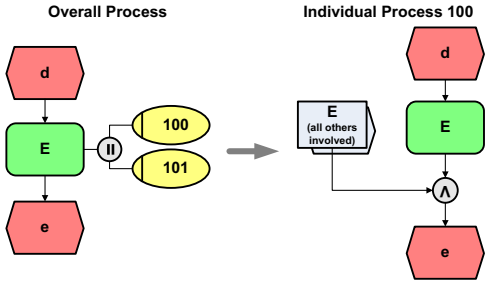

Figure 12: Parallel execution

join is introduced directly after the function, further connectors have no influence on this scenario.

Joined performance

If a function must be performed by several organisational units together, e.g., if the function is a meeting, no organisational unit can start the function without the others. For that reason the synchronisation must already be done before the start of the function. Thus introducing the AND-join-connector is required directly before the function. As according to the eEPC-rules it is not allowed that a process interface is directly followed by a function, an additional event must be introduced between the synchronisation connector and the also newly introduced process interface. The process interface depicts the previous tasks of all others involved executing the particular function and should be named accordingly. The event depicts that all others have finished their previous tasks and are ready to start performing the particular function and should be named accordingly as well (see Figure 13).

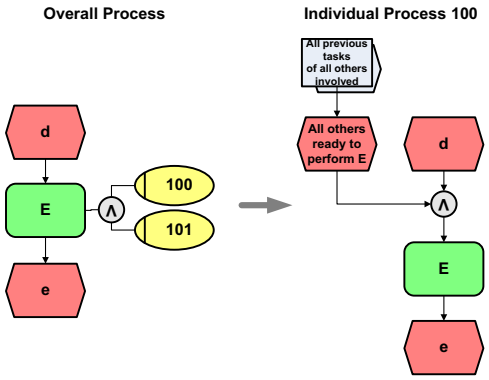

Figure 13: Joined execution

Sequential performance

The third possible scenario for multiple organisational units involved in the execution of a function is that the organisational units execute the function in a sequential order. E.g., it might be required that a manager signs the invoice release after it was processed by a clerical assistant. Probably this situation would be most obvious if the particular parts are modelled by separate functions, each executed by the particular organisational unit. However, if the individually tasks are joined into one function, a synchronisation must be introduced. The synchronisation depends on the position of the particular task within the function. The contribution of the regarded organisational unit can be the first to be executed, it can be executed in between the contributions of others, or it can be the last to

be executed.

If the organisational unit contributes first, the function in the individual process can start directly after the previous event has occurred, but the end-event cannot occur before all others have finished their contribution. Thus a synchronisation must take place before the end-event occurs – identically like in the parallel execution scenario. If the organisational unit contributes last, it cannot start performing the function before all others have finished their portion. A synchronisation with the others has to be performed before the regarded organisational unit starts – similar to the joined execution scenario. In this scenario, however, the process interface should be called according to the regarded function.

If the regarded organisational unit has to contribute in between, obviously it has to wait until other organisational units have finished their tasks. After performing the task, again some other organisational units have to contribute before the end-event of the function signalises the completion of the function (see Figure 14). Thus, this situation is a combination of the two other scenarios – or more correct, the two scenarios above are just specialised scenarios in which no others have to perform tasks before or afterwards.

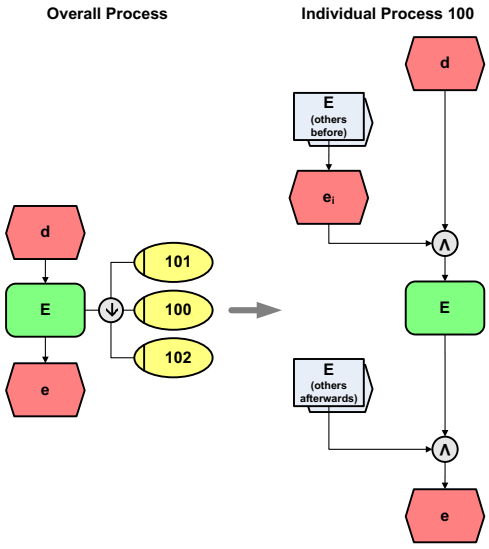

Figure 14: Sequential execution

3.3 Example

To conclude this section we will give a short example how individual process models can be derived. Therefore we use the simple invoice verification process from Figure 1. First, one of the three Organisational Units *North-East*, *West*, or *South* has to process the

invoice. Each of these organisational units can also perform a automatic invoice release. However, a manual invoice release can only be performed by the organisational unit *South*. Afterwards also a Manager has to approve the manual invoice release.

Exemplary the derived individual processes for *West* is depicted in Figure 15. *West* can process the invoice verification and the automatic invoice release, i.e. the particular functions and the surrounding events as well as arcs and connectors in between are copied to the individual process. If the invoice requires a manual release, it must be handed over to *South*, depicted by a new process interface. The function *Process Invoice* can also be executed by *North-East* and *South*, i.e. *West* must be able to pick up the process from these organisational units in order to release it automatically. This is depicted by the process interface in parallel to the *Process Invoice*-function. The two other organisational units are also able to perform the automatic release. That means, after processing the invoice *West* has to decide if it continues with the automatic release or if it leaves that task to one of the other organisational units. Also note that unnecessary connectors as e.g. the one between the function *Relesase Invoice automatically* and the event *Payment must be effected* are removed from the individual model.

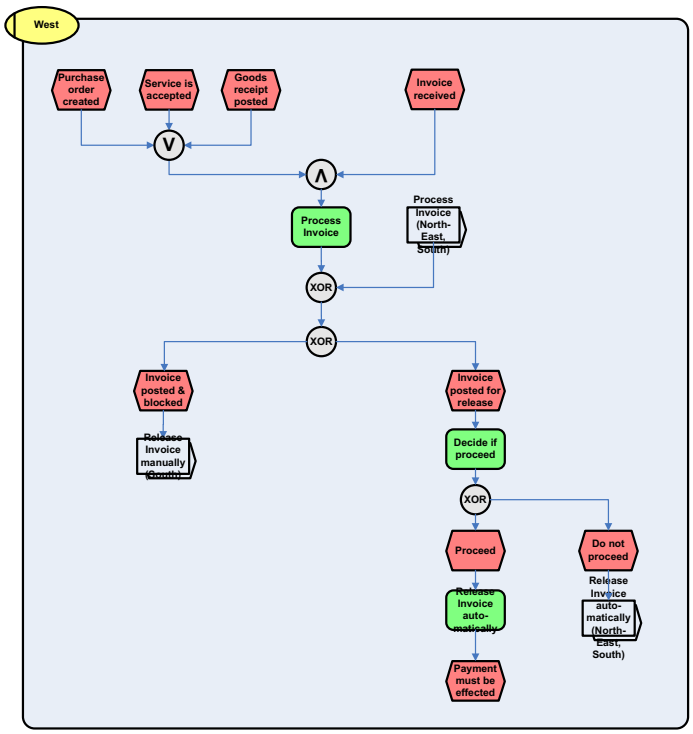

Figure 15: Individual Processes for the Organisational Unit *West*

The manager's process was already depicted in Figure 6. His only task is to agree to the manual invoice release. Therefore he has to wait for processed invoices requiring a manual release, which is depicted by the upper process interface in its individual process. However, he can only agree to the manual release after *South* has performed its contribution to the function. This is also depicted in the process by the second added process interface.

4 Summary and Outlook

Multi-perspective modelling results into fairly complex models although for each user only a subset of these models is relevant. Tailoring mechanisms hiding irrelevant model parts were therefore developed in previous research. These approaches however lack of visibility of the overall business process which is one of the most important ideas behind business process management and modelling. With process interfaces EPCs already provide a construct for keeping this overall process awareness.

Within our approach, we provide a mechanism to introduce such process interfaces quasi automatically. Besides regarding organisational breaks it is also required to distinguish between alternative and concurrent collaboration. If it is possible that a task can be performed by organsiational units alternatively, it must be analysed if additional organsiational breaks can occur. If it is required that organisational units perform a task together their processes must be synchronised. We even distinguished between joined, parallel, and sequential collaboration in a task and also provided a notation to depict these collaborations in general EPCs.

Of course the described approach is limited in several aspects which have to be considered in further research. As already depicted we focus on the involvement of organisational units within a process. However individual models might be interesting for other aspects as well, e.g. for products or locations. Theoretically individual models could be created for every attribute assigned to functions. Further on the approach neglects the hierarchy between organisational units. E.g. an individual process model for an organisational unit should probably include all functions that are assigned to its subordinated organisational units. It might also be interesting to analyse interdependencies between such hierarchies and the introduced notion of collaboration. Also dependencies between assignments of organisational units to functions within a single process instance are neglected, i.e. it might be required that an organisational unit performing a certain function also performs other functions in the same process instance. E.g. the ARIS Process Performance Manager offers the opportunity to specify the release of used resources. The consideration of such dependencies would reduce the individual model size and increase the relevance of the model even further. On the other hand it might also result into multiple versions of the same process for the same organisational unit (which creates further demands on modelling tools). Overall the suggested approach results into an increased complexity of the model base. But it reduces the complexity of the models presented to users and therefore facilitates their usability.

References

[BDFK03] J. Becker, P. Delfmann, T. Falk, and R. Knackstedt. Multiperspektivische ereignisgesteuerte Prozessketten (in German). In M. Nüttgens and F.J. Rump, editors, *EPK 2003: Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, pages 45–60, Bamberg, October 2003.

[BDKK02] J. Becker, P. Delfmann, R. Knackstedt, and D. Kuropka. Konfigurative Referenzmodellierung (in German). In J. Becker and R. Knackstedt, editors, *Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung*, pages 25–144. Heidelberg, 2002.

[BRS95] J. Becker, M. Rosemann, and R. Schütte. Grundsätze ordnungsmäßiger Modellierung (in German). *Wirtschaftsinformatik*, 37(5):435–445, 1995.

[DCRvdA05] A. Dreiling, M. Chiang, M. Rosemann, and W. M. P. van der Aalst. Towards an Understanding of Model-Driven Process Configuration and its Support at Large. In *Eleventh Americas Conference on Information Systems*, pages 2084–2092, Omaha, NE, 2005.

[HC93] M. Hammer and J. Champy. *Reengineering the corporation: A manifesto for business revolution*. Nicholas Brealey Publishing Allen & Unwin, London St Leonards, N.S.W, 1993.

[KKS04] R. Klein, F. Kupsch, and A.-W. Scheer. Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten (in German). Technical Report 178, Institut für Wirtschaftsinformatik, Saarbrücken, November 2004.

[KT98] G. Keller and T. Teufel. *SAP R/3 Process-oriented Implementation: Iterative Process Prototyping*. Addison Wesley Longman, Harlow, England Reading, Ma., 1998.

[LSS94] O. I. Lindland, G. Sindre, and A. Solvberg. Understanding Quality in Conceptual Modeling. *IEEE Software*, 11(2):42–49, March 1994.

[MG75] D. Miller and L. A. Gordon. Conceptual Levels and the Design of Accounting Information Systems. *Decision Sciences*, 6:259–269, April 1975.

[PW05] M. Price and J. Works. Balancing Roles and Responsibilities in Six Sigma. 2005. http://finance.isixsigma.com/library/content/c040211a.asp.

[RA05] M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 2005. (to appear, also available via BPMCenter.org).

[Ros96] M. Rosemann. *Komplexitätsmanagement in Prozessmodellen: methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung (in German)*. Wiesbaden, 1996.

[Ros03] M. Rosemann. Preparation of process modeling. In J. Becker, M. Kugeler, and M. Rosemann, editors, *Process Management: A Guide for the Design of Business Processes*, pages 41–78. Springer Verlag, Berlin, Heidelberg, New York, 2003.

[RSD05] M. Rosemann, A. Schwegmann, and P. Delfmann. *Vorbereitung der Prozessmodellierung (in German)*, pages 50–107. Springer, 2005.

[RzM97] M. Rosemann and M. zur Mühlen. Modellierung der Aufbauorganisation in Workflow-Management-Systemen: Kritische Bestandsaufnahme und Gestaltungsvorschläge (in German). In S. Jablonski, editor, *EMISA-Fachgruppentreffen 1997*, pages 100–118, Darmstadt, 1997.

[Sch94a] A.-W. Scheer. ARIS Toolset: A software product is born. *Information Systems*, 19(8):607–624, December 1994.

[Sch94b] A.-W. Scheer. *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.

[Sch00] A.-W. Scheer. *ARIS - Business Process Modeling*. Springer, Berlin New York, 3rd edition, 2000.

Einfache EPK-Semantik durch praxistaugliche Stilregeln

Volker Gruhn, Ralf Laue
{gruhn,laue}@ebus.informatik.uni-leipzig.de
Lehrstuhl für Angewandte Telematik und E-Business*
Universität Leipzig, Fakultät für Informatik

Abstract: Bekannte Ansätze zur Semantikdefinition von EPKs beschreiten zwei grundsätzlich unterschiedliche Wege, um die bestehenden Probleme (insbesondere mit nichtlokalen Konnektoren) zu lösen: Entweder die Klasse der „gültigen“ EPKs wird stark eingeschränkt oder ein komplexer Algorithmus berechnet die Semantik (oder stellt fest, dass für die vorliegende EPK keine vernünfige Semantik existiert).

Wir versuchen, einen Mittelweg zu finden und fragen, ob es wenige einfache Stilregeln für EPKs gibt, die eine einfache und eindeutige Semantikdefinition ermöglichen. Die Stilregeln sollen nicht unnötig streng sein. d.h. sie sollen möglichst viele in der Praxis vorhandenen Modelle abdecken.

Wir schlagen einige einfache Stilregeln vor, die die o.g. Anforderungen erfüllen. An 285 EPK-Modellen, die wir aus verschiedensten Quellen gewonnen haben, wurde geprüft, ob das Modell die Regeln einhält. Diese Untersuchung zeigte, dass in den Fällen, in denen die Stilregeln verletzt waren, tatsächlich auch das Modell einer Verbesserung bedurfte.

1 Einführung

Ereignisgesteuerte Prozessketten (EPKs) wurden ursprünglich als eine nicht vollständig formalisierte Notation entwickelt, um Geschäftsprozesse zu modellieren. Dies reicht zwar aus, um Prozesse zu dokumentieren und über die Modelle zu diskutieren, für eine automatische Verarbeitung von EPK-Modellen (etwa in Werkzeugen zur Simulation oder Verifikation) muss jedoch eine formale Semantik der Modelle definiert werden.

Hierfür gibt es verschiedene Ansätze, die wir in Abschnitt 2 besprechen. Dort stellen wir auch die Probleme dar, die es bei der Definition einer EPK-Semantik gibt und geben an, wie diese von den verschiedenen Autoren gelöst wurden.

Im Abschnitt 3 werden die bekannten Ansätze bewertet. Insbesondere stellen wir die Frage, wie gut die verschiedenen Ansätze mit den in der Praxis vorhandenen (mitunter wenig strukturierten) Modellen arbeiten können. In Abschnitt 4 schlagen wir Stilregeln für „gut modellierte EPKs“ vor. EPKs, die diesen Regeln folgen, lassen sich in Petri-Netze übersetzen, was kurz in Abschnitt 6 besprochen wird. In Abschnitt 5 zeigen wir häufige Fehler,

*Der Lehrstuhl für Angewandte Telematik und E-Business ist ein Stiftungslehrstuhl der Deutschen Telekom AG

die zur Verletzung der Stilregeln führen und deren (automatisch durchführbare) Korrektur. Im Abschnitt 7 werten wir 285 aus den verschiedensten Quellen gesammelte EPKs aus und überprüfen an diesen die Praxistauglichkeit unserer vorgeschlagenen Stilregeln.

2 Semantikdefinitionen für EPKs und auftretende Probleme

EPKs wurden ursprünglich eingeführt und verwendet, ohne ihre Semantik formal zu definieren. Demgegenüber bieten Petri-Netze, deren Semantik klar definiert ist, ebenso die Möglichkeit, Abläufe mit Parallelität und Entscheidungsknoten zu beschreiben. Die große Zahl bekannter Forschungsergebnisse aus dem Bereich der Petri-Netze sowie die gute Werkzeug-Unterstützung etwa zur Simulation von Petri-Netzen legen es nahe, zur Definition der Semantik einer EPK diese in ein Petri-Netz zu übersetzen.

Solche Übersetzungen wurden von verschiedenen Autoren vorgeschlagen, darunter van der Aalst[Aal99], Chen/Scheer[CS94], Langner, Schneider und Wehler[LSW97a, LSW97b] und Dehnert/van der Aalst[DA04]. Bei diesen Übersetzungen müssen jeweils Einschränkungen gemacht werden, die wir in den folgenden Unterabschnitten besprechen werden.

Einen anderen Weg beschreiten die Autoren Kindler und Cuntz[Kin04, Cun04, CK04]. Sie beschreiben die möglichen Abläufe eines EPK-Modells mit Hilfe eines Transitionssystems und geben ein Verfahren an, mit dem dessen Transitionsrelation bestimmt werden kann. Dieses Verfahren ist im dem Programm EPCtools implementiert, das zudem einige algorithmische Tricks benutzt, um die Berechnung auch für große EPKs effektiv ausführen zu können.

Der Grund dafür, dass über Jahre hinweg immer wieder neue Vorschläge zur Definition einer Semantik gemacht wurden, liegt im Wesentlichen in den Problemen begründet, die in den beiden folgenden Unterabschnitten besprochen werden: der Nichtlokalität von Verknüpfungskonnektoren und der fehlenden Unterstützung mehrfacher Instanziierung.

2.1 Probleme durch nichtlokale Konnektoren

Abbildung 1 zeigt ein typisches OR-Konstrukt. Es könnte einer EPK, die die Auswahl von Bewerbern für eine Stelle eines wissenschaftlichen Mitarbeiters beschreibt, entnommen sein. Nach dem OR-Split werden eine, zwei oder alle der dargestellten Aktivitäten parallel ausgeführt, und der Ablauf wird nach dem OR-Join erst fortgesetzt, wenn alle begonnenen Aktivitäten beendet sind. Bei einer Übersetzung der EPK in ein Petri-Netz ergibt sich nun die Frage, wie der OR-Join-Konnektor in ein Petri-Netz-Konstrukt zu übersetzen ist. Das Problem hierbei ist, dass am OR-Join nicht bekannt ist, wie viele parallele Abläufe am OR-Split gestartet wurden, d.h. auf wie viele vom OR-Split gesendete Tokens gewartet werden soll. Diese Frage lässt verschiedene Antwortmöglichkeiten zu, die unter anderem in [Aal99], [Rit00] und [WEAtH05] diskutiert werden. Meist wird die Frage so beantwortet, dass der OR-Join erst dann Kontrollfluss-Tokens („Prozessordner“) weiterreicht, wenn an einem ankommenden Zweig ein Kontrollfluss-Token anliegt und für alle anderen

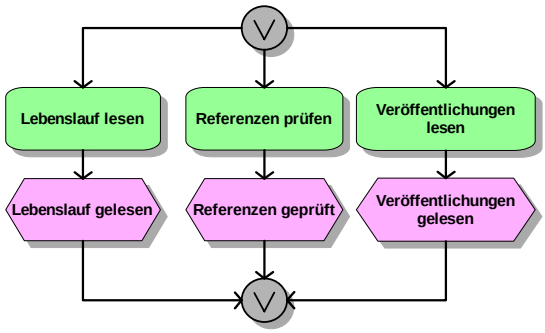

Abbildung 1: OR-Konstrukt

ankommenden Zweige bestimmt werden kann, ob noch weitere Tokens zu erwarten sind. Diese Entscheidung erfordert aber in der Regel nicht nur die Kenntnis der lokalen Situation direkt vor dem OR-Join, sondern eine Analyse des gesamten EPK-Modells. Daher wird der OR-Join-Konnektor als *nichtlokaler Konnektor* bezeichnet.

Eine ähnliche Situation finden wir bei XOR-Join-Konnektoren, deren Zweck darin besteht, alternative Kontrollflüsse zusammenzuführen. Liegt ein Token an genau einem der Eingänge eines XOR-Joins an, wird es an den Ausgang des XOR-Joins weitergeleitet. Wie die Semantik eines XOR-Joins ist, wenn an mehr als einem Eingang Kontrollfluss-Tokens eintreffen, wird unterschiedlich beantwortet. Während bei [Aal99] und [DA04] in diesem Falle die ausgehende Kontrollflusskante mehrfach durchlaufen wird, gehen andere Ansätze [ADK02, Kin04, Cun04] davon aus, dass der Ablauf blockiert wird. Folgt man dieser Interpretation, erhält auch der XOR-Join eine nichtlokale Semantik, da vor einem Weitergeben von Tokens stets zu prüfen ist, an welchen Eingängen noch Tokens ankommen könnten.

[ADK02] liefert das zentrale Ergebnis der Betrachtung nichtlokaler Konnektoren: Es ist unmöglich, eine befriedigende Semantik für EPKs anzugeben, die mit der informellen Semantik (mit nichtlokalen Konnektoren) übereinstimmt. Dies wird an einem Gegenbeispiel, bei dem zwei XOR-Joins jeweils darauf warten, dass der andere zuerst schaltet, gezeigt.

Für die Definition einer Semantik nichtlokaler Konnektoren wurden verschiedene Ansätze vorgeschlagen.

[Aal99] betrachtet nur solche EPKs, die keine OR-Joins enthalten und geht außerdem davon aus, dass XOR-Joins mehrfach schalten, wenn an mehreren eingehenden Kontrollflusskanten Kontrollfluss-Tokens ankommen. Dadurch gibt es keine nichtlokalen Konnektoren mehr, und das EPK-Modell lässt sich leicht in ein Petri-Netz übersetzen.

[LSW97a] und [CS94] stellen zusätzliche Forderungen zur Wohlgeformtheit von EPKs, insbesondere zur sauberen Verschachtelung von Prozessblöcken. Für Modelle, die diese erfüllen, ist eine Übersetzung von EPKs in (markierte) Petri-Netze möglich.

Einen grundsätzlich anderen Weg beschreiten [Kin04] und [Cun04]. Hier untersucht ein relativ komplexer Algorithmus alle möglichen Abläufe einer EPK, um deren Semantik zu

berechnen. Ist diese Berechnung erfolgreich, so stimmt sie mit der intuitiven Vorstellung einer EPK-Semantik mit nichtlokalen Konnektoren überein. Es ist aber durchaus auch möglich, dass der Algorithmus das Resultat „keine Semantikdefinition möglich“ liefert (was nach [ADK02] auch immer so sein muss).

[WEAtH05] behandelt das Problem nichtlokaler OR-Konnektoren im Kontext der Sprache YAWL[AH02], die Ergebnisse lassen sich auch auf EPKs übertragen. Hier werden Reset-Netze, eine spezielle Variante von Petri-Netzen, als formale Grundlage benutzt. Eine Entscheidung, ob noch weitere Tokens eintreffen können, wird durch Rückwärtssuche im Zustandsraum getroffen.

Abweichend von diesen Ansätzen, betrachtet [DA04] OR- und XOR-Joins als Elemente mit lokaler Semantik. Obwohl dies - wie auch einer der Autoren von [DA04] an anderer Stelle[Aal99] schrieb - nicht den üblichen Vorstellungen einer EPK-Semantik entspricht, wurde diese Semantik erfolgreich angewendet, indem während der Analyse der EPK zusätzliche Informationen vom Modellierer erfragt wurden[vDAV05].

2.2 Probleme durch die Blockierung des Tokenflusses

Die üblichen EPK-Semantiken erlauben nicht die mehrfache Instanziierung von Ereignissen oder Funktionen.[1] Wird wie in [Rum99] der Zustand einer EPK durch die Zahl der Tokens („Prozessmappen“) auf den einzelnen Modellelementen (Ereignissen, Funktionen, Prozessegweisern und Konnektoren) definiert, kann es dennoch vorkommen, dass z.B. ein Kontrollfluss-Token eine Funktion erreicht, die bereits aktiv ist. Wie sich das Modell in diesem Falle verhalten soll, ist unklar.

Analoge Überlegungen gelten, wenn wir, wie in [Kin04] vorgeschlagen, einen Zustand einer EPK durch die Zahlen der Tokens auf den *Kontrollflusskanten* definieren. In der Regel wird hier davon ausgegangen, dass eine Kontrollflusskante, die schon durch ein Token belegt ist, „blockiert“, d.h. keine weitere Tokens annehmen kann.

Ausführlich wird diese Frage in [Cun04] analysiert, wo auch eine alternative Semantikdefinition (die eine Belegung einer Kontrollflusskante mit mehreren Tokens erlaubt) besprochen wird.

3 Diskussion der vorhandenen Lösungsansätze

Die bekannten Lösungsvorschläge für die genannten Probleme lassen sich grob in zwei Klassen einteilen.

In die eine Klasse fallen die Ansätze, die die „gültigen“ EPKs durch zusätzliche Regeln für Wohlgeformtheit beschränken. So betrachtet [Aal99] nur EPKs ohne OR-Join. [LSW97a] und [CS94] verlangen strukturierte EPK-Modelle (zu jedem Split-Konnektor gehört ein

[1] [MNN05] schlägt zwar vor, EPKs um zusätzliche Notationselemente für mehrfache Instanziierung zu erweitern, definiert jedoch dazu keine formale Semantik.

Join-Konnektor gleichen Typs, die Split/Join-Konstrukte sind sauber verschachtelt, außerdem fordert [LSW97a] zusätzliche Eigenschaften von per XOR-Konnektoren modellierten Iterationen).

Bei der Bewertung dieser Ansätze teilen wir die Einschätzung aus [vDAV05]: Sie sind von einem theoretischem Standpunkt aus interessant und wichtig, aus der Sicht eines Praktikers aber oft weniger nützlich. Die Klasse der betrachteten EPKs wird durch zusätzliche Wohlgeformtheits-Regeln eingeschränkt, die vorrangig mit dem Ziel formuliert wurden, eine elegante Übersetzung wohlgeformter EPKs in Petri-Netze (oder andere Formalismen) zu erreichen. Viele in der Praxis modellierte EPKs sind nach diesen restriktiven Regeln nicht wohlgeformt und können demnach nicht übersetzt werden, selbst wenn Praktiker diese EPKs problemlos verstehen und einsetzen können.

Die Arbeiten [Kin04] und [WEAtH05][2] fallen in die andere Klasse. In beiden Fällen wird ein komplexer Algorithmus bemüht, um für *jede* denkbare EPK eine Semantik zu bestimmen (oder - wenn dies nicht möglich ist - festzustellen, dass es keine sinnvolle Semantik gibt). Selbstverständlich ist das die bestmögliche Lösung aus theoretischer Sicht. Wir meinen allerdings, dass man für praktische Belange auf die Verwendung solch komplexer Algorithmen verzichten kann, da es eigentlich gar nicht wünschenswert ist, für jede (auch noch so wirr modellierte) EPK eine Semantik zu bestimmen. Wenn (wie in [Cun04] erwähnt) ein Computerprogramm 20 Minuten benötigt, um eine Entscheidung über die Bedeutung einer EPK zu treffen, ist es sehr unwahrscheinlich, dass dieses Modell einem der Hauptanliegen der Geschäftsprozessmodellierung - der Möglichkeit, über ein gezeichnetes Modell zu diskutieren - gerecht werden kann. Daher sollte man ein solches Modell in jedem Falle durch ein einfacher modelliertes ersetzen.

Um die Nachteile beider Klassen zu vermeiden, haben wir uns die Frage gestellt, ob es möglich sei, wenige „Stilregeln für gut modellierte EPKs“ zu finden, die folgende Bedingungen erfüllen:

1. EPKs, die in der Praxis vorkommen und über deren Semantik sich Praktiker einig sind, sollten nicht durch zu strenge Einschränkungen ausgeschlossen werden.
2. Es soll möglich sein, für EPKs, die den Stilregeln entsprechen, eine Semantik (etwa als Übersetzung in Petri-Netze) anzugeben.
3. Die Stilregeln sollen Modellierern, die mit EPKs vertraut sind, nicht „unnatürlich“ vorkommen. Statt dessen sollen sie möglichst ohnehin schon (unbewusst) angewendet werden.
4. Die Einhaltung der Regeln soll leicht und automatisiert überprüft werden können.

Modelle, die nicht den Stilregeln entsprechen, nennen wir *unzulässig*. Solche Modelle sollen unserer Auffassung nach nicht benutzt werden. Insbesondere werden wir auch auf jeglichen Versuch verzichten, die Frage nach der Semantik für diese Modelle zu beantworten. Dass wir mit diesem Ansatz trotzdem nahezu keine Probleme bei in der Praxis anzutreffenden EPKs haben, zeigt unsere Untersuchung in Abschnitt 7.

[2] [WEAtH05] behandelt keine EPKs, sondern OR-Joins in der Sprache YAWL, die Ergebnisse lassen sich aber auf EPKs übertragen.

Im nächsten Abschnitt beschreiben wir einen Vorschlag für solche Stilregeln.

4 Stilregeln

4.1 Blockierung des Tokenflusses

EPKs sehen „von Haus aus“ keine mehrfache Instanziierung von Funktionen und Ereignissen vor. Eine mögliche Definition eines Zustands der EPK sagt aus, dass der Zustand durch die Zahlen der Tokens auf den Kontrollflusskanten bestimmt ist, ohne diese Zahl zu beschränken[Cun04]. Ist es dann in einem Modell möglich, dass ein Token eine Kontrollflusskante erreicht, die bereits durch ein Token belegt ist, lässt sich daraus (wenn beide Tokens synchron „weiterwandern“) eine Situation ableiten, in der nachfolgende Funktionen oder Ereignisse von beiden Tokens erreicht werden können. Dies würde einer mehrfachen Instanziierung entsprechen. Es ist also die Schlussfolgerung naheliegend, dass dieses Modell fehlerhaft ist.

Als erste Stilregel fordern wir demnach, dass eine „gut modellierte“ EPK garantiert, dass nie mehrere Tokens zugleich eine Kontrollflusskante erreichen. Andere EPKs betrachten wir als unzulässig. Die genannte Stilregel kann leicht mit einem Petri-Netz-Analysetool überprüft werden, wenn die EPK in ein Petri-Netz übersetzt wurde.

4.2 XOR-Joins

Verschiedene wissenschaftliche Veröffentlichungen zur EPK-Syntax wie [NR02] und [Kin04] gehen von einer nichtlokalen Semantik von XOR-Joins aus: Ein XOR-Join reicht Tokens genau dann weiter, wenn an genau einer eingehenden Kontrollfluss-Kante ein Token anliegt und vor Durchlaufen des XOR-Joins *an keiner weiteren eingehenden Kante weitere Tokens eintreffen können*. Treffen also mehrere Tokens am XOR-Join ein, blockiert dieser.

Unserer Erfahrung nach entspricht dieses Blockieren jedoch nicht den Vorstellungen, die Modellierer aus der Praxis vom Verhalten des XOR-Joins haben. Wir wählen daher die auch in [Aal99] und [DA04] vorgeschlagene *lokale* Semantik für XOR-Joins: Sobald ein Token den XOR-Join erreicht, wird dieses weitergeleitet. Das spiegelt den „Normalfall“ wider, den ein Modellierer bei der Benutzung eines XOR-Joins im Sinn hat: dass er sich darauf verlassen kann, dass ohnehin nur an einer eingehenden Kontrollflusskante des XOR-Joins ein Token eintrifft.

Ist nun das Modell so gestaltet, dass (mit dieser *lokalen* Semantik für XOR-Joins) an mehreren eingehenden Kontrollflusskanten Tokens an einem XOR-Join ankommen können, gelangen diese (da der XOR-Join die Tokens ja sofort weiterleiten darf) beide an die vom XOR-Join abgehende Kontrollflusskante. Gerade das führt aber nach Abschnitt 4.1 dazu, dass wir das Modell als unzulässig betrachten.

Modelle mit XOR-Joins mit unklarer (nichtlokaler) Semantik wie der in [ADK02] vorgestellte „Teufelskreis“ (vicious circle) werden durch auf diese Weise als unzulässige EPKs eingestuft, da (wenn XOR-Joins immer Tokens weiterleiten dürfen) mehrere Eingänge eines anderen XOR-Joins belegt werden können.

Bei der Untersuchung vorhandener EPK-Modelle (siehe Abschnitt 7) fanden wir *kein* Modell, dass inhaltlich korrekt war und nach dem beschriebenen Ansatz unnötigerweise als unzulässig eingestuft würde. Keines der von uns untersuchten EPK-Modelle macht bewusst von der Deutung Gebrauch, dass der Kontrollfluss an XOR-Joins, die von mehr als einem Token erreicht werden, blockiert wird. (Dies wäre auch ohnehin eine schlechte Modellierung.) Mehr noch: Wenn immer im Modell mehr als ein Token am XOR-Join ankommen konnte (wiederum unter Annahme einer *lokalen* Semantik für alle anderen XOR-Joins), zeigte eine genauere Analyse, dass das Modell tatsächlich fehlerhaft war. Diese beiden Tatsachen belegen unserer Ansicht nach die Berechtigung, eine lokale Semantik für XOR-Joins zu betrachten und wie beschrieben gewisse Modelle als unzulässig einzustufen.

4.3 OR-Join-Konnektoren

OR-Joins sind das EPK-Notationselement, über dessen semantische Bedeutung am meisten diskutiert wurde[Aal99, LSW97a, Rit00, WEAtH05].

Die informelle Semantik für OR-Joins, die eindeutig einem OR-Split zugeordnet sind, lautet: „Warte, bis alle vom OR-Split losgeschickten Tokens eingetroffen sind und leite das Token dann an die abgehende Kontrollflusskante weiter“. Es ist dann u.a. ein Weg zu finden, wie der OR-Join erfährt, auf welche Tokens er warten muss. [LSW97a] beschreibt hierfür eine Lösung mit Hilfe boolescher Netze (also 0/1-markierter Petri-Netze). Der OR-Split sendet mit 1 markierte Tokens auf den Kontrollflusskanten, auf denen Kontrollfluss stattfinden soll und mit 0 markierte Tokens auf den anderen „nicht aktivierten“ Kanten. Der OR-Join wartet nun, bis an allen eingehenden Kontrollflusskanten ein Token eintrifft. Ist mindestens eines davon ein 1-Token , dann wird der Kontrollfluss an die vom OR-Join abgehende Kante weitergeleitet.

Unglücklicherweise ist für diese elegante Lösung ein hoher Preis zu bezahlen: [LSW97a] betrachtet nur EPKs, die sehr strengen Wohlgeformtheits-Kriterien entsprechen. Ein erheblicher Teil der in der Praxis vorkommenden EPKs mit OR-Joins entspricht diesen Kriterien nicht. Somit ist [LSW97a] mit seinen strengen Kriterien ein gangbarer Weg, wenn man Regeln für Modellierer in einem neuen Projekt festschreiben kann. Liegen aber (wie oft anzutreffen) schon EPKs vor, erfüllen diese meist nicht die Kriterien und können folglich nicht analysiert werden.

Ausgehend von den von uns untersuchten EPKs haben wir uns nun die Frage gestellt, ob auch weniger strenge (und damit mehr praxisnahe) Stilregeln für EPKs ausreichen, um den Ansatz von [LSW97a], boolesche Netze für die Modellierung von OR-Konstrukten zu verwenden, zu ermöglichen.

Tatsächlich war es möglich, solche Stilregeln aufzustellen. Hierzu definieren wir zunächst,

was wir unter einem *wohlstrukturierten Konstrukt* verstehen wollen. (Wir abstrahieren hier von Funktionen und Ereignissen in der EPK, da die kritischen Elemente allein die Konnektoren sind. Funktionen und Ereignisse sind gemäß den in [NR02] gestellten Forderungen einzusetzen.)

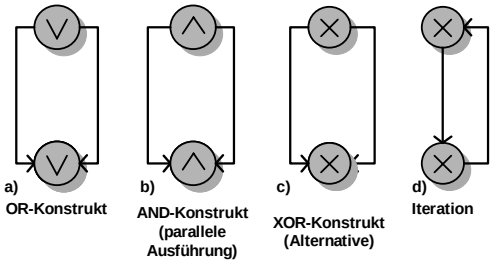

Abbildung 2: Workflow-Konstrukte

Definition 1 *(wohlstrukturierte Konstrukte):*

1. *Die in Abb. 2 gezeigten Workflow-Konstrukte sind wohlstrukturiert. In den Abb. 2 a)-c) sind auch mehr als zwei Kontrollflusskanten zwischen Split- und Join-Konnektor zulässig.*
2. *Wird in eine Kante eines wohlstrukturierten Konstruktes ein weiteres wohlstrukturiertes Konstrukt eingesetzt (siehe Abb. 3), ist das erhaltene Konstrukt wiederum wohlstrukturiert.*
3. *Wird ein „Aussprung“ (siehe Abb. 4, der XOR-Konnektor kann auch durch OR oder AND ersetzt werden) in eine Kante eines wohlstrukturierten Konstruktes eingesetzt, ist das erhaltene Konstrukt wiederum wohlstrukturiert.*
4. *Wird eine Kante eines wohlstrukturierten Konstrukts unterbrochen und durch ein Endereignis beendet (siehe Abb. 5), ist das erhaltene Konstrukt wiederum wohlstrukturiert, wenn der erhaltene Graph noch zusammenhängend ist.*

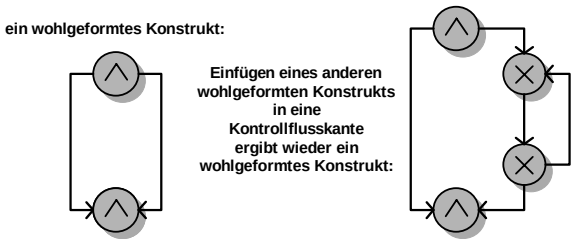

Abbildung 3: Definition, Regel 2

Die ersten beiden Regeln gewährleisten wie auch bei [LSW97a] gefordert, dass zu jedem Join-Konnektor ein zugehöriger Split-Konnektor gleichen Typs gehört. Die Regeln 3 und

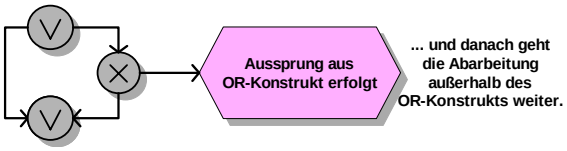

Abbildung 4: Definition, Regel 3

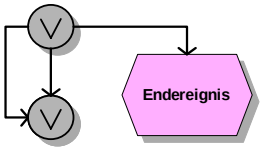

Abbildung 5: Definition, Regel 4

4 tragen der in der Praxis verbreiteten Gewohnheit Rechnung, den Kontrollfluss durch „Aussprünge“ aus einem Split/Join-Konstrukt zu unterbrechen oder durch ein Endereignis ganz abzubrechen.[3]

Wir betrachten nun nur solche EPKs als *zulässig*, für die alle OR-Joins ein wohlstrukturiertes Konstrukt entsprechend Def. 1 beenden. Insbesondere muss dieses Konstrukt dann mit einem OR-Split beginnen, d.h. jedem OR-Join muss ein OR-Split zugeordnet sein. Die Klasse der lt. unserer Definition zulässigen EPKs ist größer als die Klasse der in [LSW97a] betrachteten EPKs, da zum einen weniger strenge Anforderungen an XOR-Konnektoren in Iterationen gestellt werden, zum anderen die Regeln 3 und 4 in Def. 1 hinzukommen. Im Abschnitt 7 werden wir sehen, dass unsere Definition für zulässige EPKs nahezu keine der von uns gesammelten 285 in der Praxis vorkommenden EPKs unnötigerweise als „unzulässig“ ausschließt.

Im Gegensatz zu den in 4.1 und 4.2 genannten Stilregeln, zu deren Test eine Verhaltensanalyse des Modells nötig ist, lassen sich die Stilregeln für OR-Konstrukte mittels statischer Analyse der EPK nachweisen. Dies kann ähnlich zu dem aus [LSW97a] bekannten Vorgehen geschehen, wobei jedoch „Aussprünge“ aus Split/Join-Konstrukten und an beliebigen Stellen erlaubte Endereignisse zusätzliche Überlegungen erfordern.

5 Häufige Fehler und deren Korrektur

[LSW97a] nennt zahlreiche Modellierungsfehler, die teilweise mittels statischer Analyse einer EPK gefunden werden können. Eine solche Analyse müssen wir nach unserem Ansatz ohnehin durchführen, um die Gültigkeit der in Def. 1 aufgeführten Regeln zu prüfen. Bei dieser Gelegenheit können einige typische Modellierungsfehler gleich erkannt und verbessert werden.

Wir haben bei den von uns gesammelten EPKs typische Fehler an OR-Joins analysiert. Oft wurden OR-Joins genutzt, wenn ein XOR- oder AND-Join angebracht gewesen wäre (siehe Abb. 6a)-c), wo wieder Funktionen und Ereignisse weggelassen sind.) Ebenso wurde oft die optionale Ausführung fehlerhaft modelliert (siehe Abb. 6d)). Natürlich ändert sich bei allen in Abb. 6 gezeigten Änderungen nicht die Semantik der EPK, vom theoretischen Standpunkt aus sind die Korrekturen sogar unnötig. Wir bezeichnen diese Modelle trotz-

[3] Man kann natürlich einwenden, dass entsprechend der genannten Gewohnheiten modellierte EPKs schlecht modelliert sind, da ein Endereignis erreicht werden kann, wenn noch Synchronisationen ausstehen. Die Zielrichtung dieser Veröffentlichung ist aber eine andere: Wir nehmen zur Kenntnis, welche Modelle in der Praxis vorhanden sind und versuchen, diese so gut wie möglich zu analysieren.

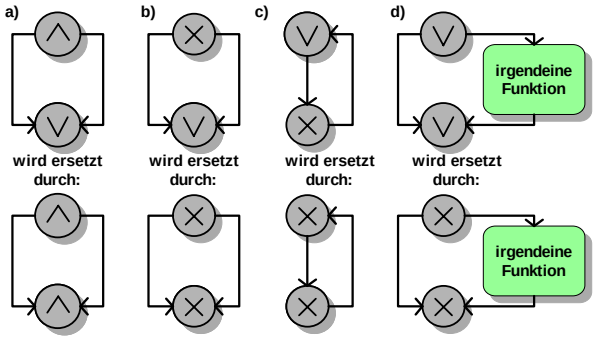

Abbildung 6: Korrektur typischer Fehler

dem bewusst als fehlerhaft, da die geänderten Modelle den zu modellierenden Sachverhalt offensichtlich besser wiedergeben. Da ein Hauptzweck von EPKs oft ist, über das Modell zu kommunizieren, sollten die „schlechten“ Konstrukte durch die besseren Varianten ersetzt werden, um die Gefahr von Missverständnissen zu verringern.

6 Übersetzung in Petri-Netze

Um nun EPKs, die unseren Stilregeln entsprechen und somit gültig sind, in Petri-Netze zu übersetzen, können wir die in [LSW97a] angegebene Übersetzung übernehmen. Zusätzlich müssen Iterationen (Abb. 2d)) entsprechend der Semantik der XOR-Konnektoren behandelt werden, und es muss dafür gesorgt werden, dass „Aussprünge“, die ein Split-Join-Konstrukt vorzeitig beenden (vgl. Abb. 4) 0-Tokens an den nachgeordneten OR-Join senden (und dieser die 0-Tokens ggf. an weitere nachgeordnete OR-Joins weiterreicht). Beide Erweiterungen des Übersetzungsalgorithmus aus [LSW97a] sind recht einfach, denn wir wissen nach der statischen Analyse zu der in Def. 1 gegebenen Stilregel, ob ein XOR-Split die Situation aus Abb. 2c) oder d) oder einen „Aussprung“ wie in Abb. 4 darstellt.

Tab. 1 fasst die wesentliche Ergänzung zu der in [LSW97a] angegebenen Übersetzung zusammen. Sie stellt für alle möglichen Split-Konnektoren dar, wie mit 0 und 1 markierte Tokens an die ausgehenden Kontrollflusskanten (symbolisiert durch die Variablen x und y) weitergeleitet werden, wenn auf der eingehenden Kontrollflusskante (symbolisiert durch die Variable z) ein mit 0 oder 1 markiertes Token eintrifft. Die Angabe „0“ oder „1“ in der entsprechenden Spalte besagt, dass ein entsprechend markiertes Token gesendet wird; ein Strich zeigt an, dass kein Token gesendet wird. Im Fall „xor-Split in Iteration“ ist die Variable x der Kontrollflusskante zugeordnet, die die Schleife beendet, y der anderen. In den Fällen „. . . -Aussprung“ ist x der Kontrollflusskante zugeordnet, die im wohlstrukturierten Konstrukt liegt, und y der Kontrollflusskante , die „herausspringt“. Alle anderen Bezeichnungsweisen und das generelle Vorgehen entsprechen dem in [LSW97a] angegebenen.

	and-Split		xor-Split		or-Split		xor-Split in Iteration		xor-Aussprung		or-Aussprung		and-Aussprung	
Abb.	2a)		2b)		2c)		2d)		4		analog 4		analog 4	
z	x	y	x	y	x	y	x	y	x	y	x	y	x	y
0	0	0	0	0	0	0	0	-	0	-	0	-	0	-
1	1	1	1 0	0 1	1 0 1	0 1 1	- 1	1 -	0 1	1 -	0 1 1	1 - 1	1	1

Tabelle 1: Schaltungen an Split-Verknüpfern

7 Untersuchung von EPK-Modellen

Die Zielstellung dieser Arbeit lag darin, Stilregeln für EPKs zu finden, die einerseits eine korrekte Übersetzung der EPKs in Petri-Netze erlauben, andererseits jedoch die Klasse der gültigen EPKs nicht so stark einschränken, dass der Ansatz für viele EPKs aus der Praxis unbrauchbar ist.

Um dies zu überprüfen, haben wir EPKs aus allen uns zugänglichen Quellen zusammengesucht, insgesamt 285 Stück. Die Quellen hierfür waren 23 Diplomarbeiten, 2 Seminararbeiten, 4 Promotionen, 5 Bücher (eines davon mit zahlreichen Beispielen des SAP-Referenzmodells), 30 wissenschaftliche Veröffentlichungen, Kursfolien einer Lehrveranstaltung zum Thema „EPKs“ sowie eines unserer eigenen Projekte, bei dem Abläufe bei einem Energieversorger modelliert wurden. Die komplette Quellenliste ist auf [Lau05] nachzulesen. EPKs mit kleinen syntaktischen Fehlern haben wir toleriert; 9 Modelle, die in unseren Quellen als „EPK“ bezeichnet wurden, hatten jedoch solch gravierende syntaktische Fehler (etwa Aktivitäten, von denen mehrere Kontrollflusskanten ausgehen), dass wir sie beim besten Willen nicht mehr als EPK ansehen konnten und sie daher für die weitere Analyse ignorieren mussten.

Für die verbleibenden 276 EPK-Modelle fanden wir folgendes heraus:

- *Keines* der Modelle benutzt bewusst die Interpretation, dass zwei an einem XOR-Join zugleich ankommende Tokens den Ablauf blockieren. Wann immer in einem Modell mehr als ein Token zugleich am XOR-Join ankommen konnte, war das Modell *klar erkennbar fehlerhaft*. Damit ist die in Abschnitt 4.2 vorgeschlagene lokale Semantik von XOR-Joins bei gleichzeitigem Verbot von EPKs, bei denen mehrere Tokens zugleich einen XOR-Join erreichen, sinnvoll.
- 190 EPKs verwendeten keine OR-Joins.
- Die verbleibenden 86 EPKs enthielten insgesamt 151 OR-Joins. 94 davon entsprachen den Stilregeln aus Abschnitt 4.3.

Das eigentlich interessante Resultat ergab sich nun bei der („in Handarbeit“ durchgeführten) Untersuchung der 57 OR-Joins, die nicht den Stilregeln aus Abschnitt 4.3 entsprachen.

Auf 45 davon traf einer der in Abb. 6 gezeigten Fälle zu, d. h. sie sollten besser durch einen anderen Join-Konnektor ersetzt werden. Wie schon erwähnt, kann diese Ersetzung automatisch erfolgen.

Für 10 weitere EPKs, deren OR-Joins nicht den Stilregeln entsprachen, ergab eine genauere Betrachtung, dass diese offensichtlich fehlerhaft waren.[4]

Wir fanden nur zwei EPKs, die nicht den Stilregeln entsprechen und trotzdem als korrekt modelliert angesehen werden könnten. Beide benutzten das in Abb. 7 gezeigte Konstrukt. Da in diesem Konstrukt ein Token am AND-Join „steckenbleibt“, wenn nur eine der Ausnahmen eintritt (was übrigens dann auch verbietet, dieses Konstrukt in einer Schleife mehrfach durchlaufen zu lassen), sollte eine solche Modellierung vermieden werden, so dass eine Einstufung des Modells als „unzulässig“ durchaus vertretbar ist.

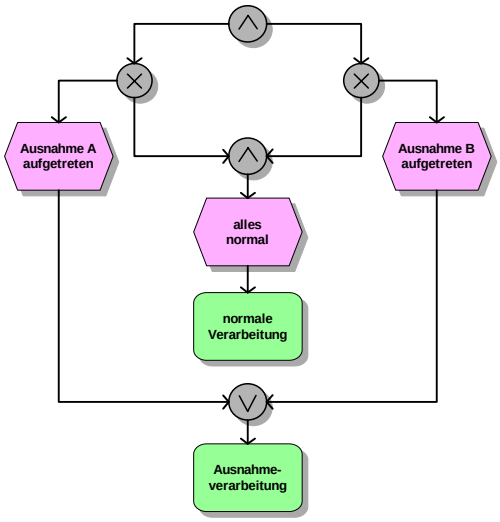

Abbildung 7: EPK, die unsere Stilregeln verletzt

8 Zusammenfassung

Die im letzten Kapitel genannten Zahlen belegen, dass unsere Stilregeln von nahezu allen in der Praxis anzutreffenden Modellen befolgt werden. Dies geschieht meist schon intuitiv, man kann die Regeln dem Modellierer aber auch mit wenigen (bewusst informell

[4] „Fehlerhaft“ heißt an dieser Stelle „inhaltlich falsch“, was nur durch Betrachtung der modellierten Geschäftsprozesse herauszufinden ist. Ein Beispiel aus einem unserer eigenen Modelle war die Modellierung eines Falles, in dem ein Stromzähler zugleich defekt und in Ordnung war. Ein anderes Beispiel eines fehlerhaften Modells war das bereits von verschiedenen Autoren untersuchte Modell „Beschaffungslogistik“[Rit00]

formulierten) Sätzen vermitteln:

1. Beachte (bei Modellen mit Zyklen), dass keine Funktion mehrfach zugleich aktiviert werden kann.
2. Beachte, dass ein XOR-Join immer nur von genau einer Seite erreicht wird.
3. Denke bei der Verwendung von OR-Konstrukten immer zunächst darüber nach, ob du nicht eigentlich „AND“ oder „XOR“ meinst. Wenn du wirklich OR-Konstrukte verwenden musst, beachte, dass innerhalb dieser Konstrukte zu jedem Split- ein Join-Konnektor gleichen Typs gehört und dass man nicht von außerhalb des OR-Split/OR-Join-Konstrukts in dieses hineinspringen kann.

Mehr noch, die Ergebnisse zeigen, dass Modelle, die den Stilregeln *nicht* entsprechen, auch tatsächlich *ausnahmslos* fehlerhaft[5] waren. Das nächste interessante Resultat war, dass sich die Mehrzahl dieser Fehler automatisch mit den in der statischen Analyse gewonnenen Erkenntnissen korrigieren lässt.

Die Abdeckung von EPKs aus der Praxis ist mit unseren Stilregeln deutlich besser als bei den in [LSW97a] genannten Regeln, jedoch kann die in [LSW97a] vorgeschlagene Übersetzung von EPKs in boolesche Netze auch für die Klasse der nach unserer Definition gültigen EPKS erfolgen.

Damit erhalten wir das positive Resultat, dass die meisten in der Praxis vorhandenen EPKs auf relativ einfache (zumindest im Vergleich zu den Algorithmen aus [CK04] oder [WEAtH05]) Weise in eine Sprache übersetzt werden können, die zur Weiterverarbeitung in Simulations- oder Model-Checking-Tools dienen kann.

Literatur

[Aal99] Wil M.P. van der Aalst. Formalization and verification of event-driven process chains. *Information & Software Technology*, 41(10):639–650, 1999.

[ADK02] Wil M.P. van der Aalst, Jörg Desel und Ekkart Kindler. On the semantics of EPCs: A vicious circle. In *EPK 2004, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, Seiten 71–79, 2002.

[AH02] Wil M.P. van der Aalst und A. Hofstede. YAWL: Yet Another Workflow Language. Bericht FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002.

[CK04] Nicolas Cuntz und Ekkart Kindler. On the semantics of EPCs: Efficient calculation and simulation. In *EPK 2004: Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings*, Seiten 7–26, 2004.

[CS94] R. Chen und A.W. Scheer. Modellierung von Prozessketten mittels Petri-Netz-Theorie. *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, (107), 1994.

[5] Hier wieder „fehlerhaft“ im weiteren Sinne: Gemeint sind inhaltlich falsche Modelle wie auch solche, für die eine klar bessere Darstellung mit gleicher Semantik existiert.

[Cun04] Nicolas Cuntz. Über die effiziente Simulation von Ereignisgesteuerten Prozessketten. Diplomarbeit, Universität Paderborn, 2004.

[DA04] Juliane Dehnert und Wil M.P. van der Aalst. Bridging The Gap Between Business Models And Workflow Specifications. *Int. J. Cooperative Inf. Syst.*, 13(3):289–332, 2004.

[Kin04] Ekkart Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In *Business Process Management*, Seiten 82–97, 2004.

[Lau05] Ralf Laue. ebus.informatik.uni-leipzig.de/∼laue, 2005.

[LSW97a] P. Langner, C. Schneider und J. Wehler. Ereignisgesteuerte Prozessketten und Petri-Netze. *Berichte des Fachbereichs Informatik der Universität Hamburg*, (106), 1997.

[LSW97b] P. Langner, C. Schneider und J. Wehler. Prozessmodellierung mit ereignisgesteuerten Prozessketten (EPKs) und Petri-Netzen. *Wirtschaftsinformatik*, 39(5):479–489, 1997.

[MNN05] Jan Mendling, Gustaf Neumann und Markus Nüttgens. Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). In *Second GI-Workshop XML4BPM XML for Business Process Management*, 2005.

[NR02] Markus Nüttgens und Frank J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In *Promise 2002 - Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen*, Seiten 64–77, 2002.

[Rit00] Peter Rittgen. Quo vadis EPK in ARIS? *Wirtschaftsinformatik*, 42(1):27–35, 2000.

[Rum99] Frank J. Rump. *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten*. B. G. Teubner Verlag Stuttgart Leipzig, 1999.

[vDAV05] Boudewijn F. van Dongen, Wil M.P. van der Aalst und H. M. W. Verbeek. Verification of EPCs: Using Reduction Rules and Petri Nets. In *CAiSE*, Seiten 372–386, 2005.

[WEAtH05] Moe Thandar Wynn, David Edmond, Wil M.P. van der Aalst und Arthur H. M. ter Hofstede. Achieving a General, Formal and Decidable Approach to the OR-Join in Workflow Using Reset Nets. In *ICATPN*, Seiten 423–443, 2005.