Mathias Weske, Markus Nüttgens (Hrsg.)

EMISA 2006

Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

**Beiträge des Workshops
der GI-Fachgruppe Entwicklungsmethoden für
Informationssysteme und deren Anwendung (EMISA)**

17.-18. Oktober 2006 in Hamburg

Gesellschaft für Informatik 2006

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-

ISSN
ISBN

Volume Editors
Prof. Dr. Mathias Weske
Hasso-Plattner-Institut für IT Systems Engineering
Universität Potsdam
Prof.-Dr.-Helmertstr. 2-3
D-14482 Potsdam
E-mail: mathias.weske@hpi.uni-potsdam.de

Prof. Dr. Markus Nüttgens
Professur für Wirtschaftsinformatik
Universität Hamburg
Fakultät für Wirtschafts-und Sozialwissenschaften
Von-Melle-Park 9
D-20146 Hamburg
E-mail: nuettgens@wiso.uni-hamburg.de

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Die EMISA'2006 ist die 27. jährliche Tagung der GI-Fachgruppe Entwicklungsmethoden für Informationssysteme und deren Anwendung. In diesem Jahr findet das Fachgruppentreffen vom 17.-18. Oktober 2006 an der Universität Hamburg statt.

Dienstbasierte Informationssysteme bilden den inhaltlichen Schwerpunkt des diesjährigen Fachgruppentreffens. Derzeit werden dienstbasierte Architekturen – Service-Oriented Architectures – als viel versprechender Ansatz angesehen, um Anwendungen schnell und kostengünstig zu entwickeln und an sich ändernde Anforderungen mit geringem Aufwand anzupassen. Um die Möglichkeiten dienstbasierter Architekturen zu realisieren bedarf es einer umfassenden methodischen Innovation. Hierzu müssen einerseits existierende Konzepte und Technologien für komponentenbasierte Ansätze weiter entwickelt werden. Andererseits strebt man eine verbesserte Flexibilität der Anwendungsarchitektur an, die nur mit neuartigen Konzepten für die Entwicklung, die Organisation und die Ausführung von Diensten realisierbar ist. Vor diesem Hintergrund beschäftigt sich das Fachgruppentreffen EMISA'2006 mit neuartigen Methoden, Konzepten und Technologien für die Entwicklung von dienstbasierten Informationssystemen.

Die eingereichten Beiträge wurden in einem Begutachtungsprozess von jeweils mindestens zwei Gutachtern bewertet. Der vorliegende Tagungsband enthält die zwölf ausgewählten Beiträge. Inhaltlich gliedern sich diese Beiträge in die vier Themenbereiche Grundlagen, Entwicklungsprozesse, Anwendungen und Geschäftsprozesse.

Für ihre Beteiligung am diesjährigen Fachgruppentreffen möchten wir uns bei den Autoren und Autorinnen bedanken, die Beiträge eingereicht haben. Bei den Mitgliedern des Programmkomitees bedanken wir uns für die sorgfältige und termingerechte Begutachtung. Unser besonderer Dank gilt den lokalen Organisatoren sowie Hilmar Schuschel für die Zusammenstellung des Tagungsbandes. Schließlich auch vielen Dank an Herrn Kuck von der Köllen Druck+Verlag Gmbh sowie an Frau Winter von der GI für die wie gewohnt gute Zusammenarbeit.

Potsdam und Hamburg, im September 2006

Mathias Weske und Markus Nüttgens

Programmkomitee

Witold Abramowicz (Poznan University of Economics, Polen)
Michael Altenofen (SAP Research)
Andreas Gadatsch (FH Bonn-Rhein-Sieg)
Jörg Desel (KU Eichstätt-Ingolstadt)
Schahram Dustdar (TU Wien)
Johann Eder (U Wien)
Fernand Feltz (CREDI Luxemburg)
Dieter Fensel (U Innsbruck)
Ulrich Frank (U Duisburg Essen)
Holger Günzel (IBM Stuttgart)
Heinrich Jasper (TU Freiberg)
Gerti Kappel (TU Wien)
Roland Kaschek (Massey University Palmerston North, Neuseeland)
Ralf Klischewski (German University in Cairo, Ägypten)
Horst Kremers (CODATA Berlin)
Dominik Kuropka (HPI, U Potsdam)
Wilfried Lamersdorf (U Hamburg)
Heinrich C. Mayr (U Klagenfurt)
Christoph Meinel (HPI, Uni Potsdam)
Markus Nüttgens (U Hamburg)
Andreas Oberweis (U Karlsruhe)
Mike Papazoglou (U Tilburg, Niederlande)
Hansjürgen Paul (IAT Gelsenkirchen)
Manfred Reichert (U Twente, Niederlande)
Wolfgang Reisig (HU Berlin)
Norbert Ritter (U Hamburg)
Gunter Saake (U Magdeburg)
Steffen Staab (U Koblenz)
Bernhard Thalheim (U Kiel)
Jari Veijalainen (U Jyväskylä, Finnland)
Gottfried Vossen (U Münster)
Mathias Weske (HPI, Uni Potsdam)
Guido Wirtz (U Bamberg)

Organisatoren

Leitung des Programmkomitees

- Mathias Weske (HPI, Universität Potsdam)

Organisationsleitung

- Markus Nüttgens (Universität Hamburg)

Inhaltsverzeichnis

Geschäftsprozesse

A Unified Formal Foundation for Service Oriented Architectures

Frank Puhlmann

Business Process Technology Group
Hasso-Plattner-Institute for IT Systems Engineering
at the University of Potsdam
D-14482 Potsdam, Germany
frank.puhlmann@hpi.uni-potsdam.de

Abstract: This paper summarizes how an algebra for mobile systems, the π-calculus, can be applied as unified formal foundation to service oriented architectures (SOA). The concepts accounted are orchestrations including data and processes, as well as choreographies consisting of interacting processes. Since SOAs incorporate agile binding of interaction partners, static process structures as found in Petri nets are not sufficient for completely representing orchestrations and choreographies. The π-calculus, in contrast, inherently supports link passing mobility required for agile interacting processes.

1 Introduction

Service oriented architectures (SOA) as introduced by Burbeck in [Bur00] consists of three major roles. A *service provider* publishes information about the services it offers at a *service broker*, where in turn they can be found by *service requesters*. Once a service requester decides to incorporate a service provider, it dynamically binds to it. While today there exist many standards from the WS-stack (e.g. [CCMS01, W3C04, BEA03]) that describe how certain aspects of a SOA can be implemented, as well as scientific research regarding specific areas (e.g. [vAW01, Mar05, SBS04]), a unified formal foundation for service oriented architectures is still missing. None of the existing standards is based on a common formal foundation, and most of the scientific approaches utilize either Petri nets or proprietary formalizations. Since this leads to a lack of common understanding and continuous research, this paper summarizes our results on investigating an algebra for mobile systems, the π-calculus [MPW92], as a unified foundation for SOA. In contrast to Petri nets [Pet62], the π-calculus inherently supports link passing mobility required for dynamic binding in agile interactions.

Figure 1 depicts link passing mobility. The left hand side shows the three different roles of a SOA, denoted as circles. A service requester (R) knows a service broker (B), denoted by the line (link) between, where the dot denotes the target. The service broker, in turn, has knowledge about a number of service providers (P). The service broker evaluates the

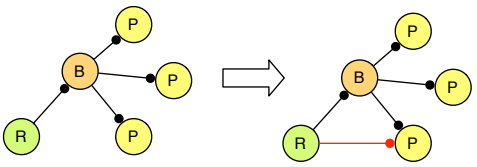


Figure 1: A service oriented architecture in π-calculus.

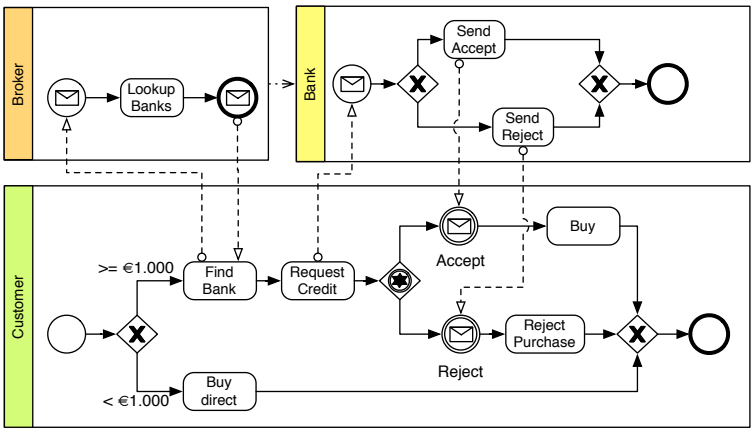


Figure 2: An example choreography in BPMN notation.

request of the service requester and returns the corresponding link. The service requester then uses this link to dynamically bind to the service provider. Hence, the link structure of the example changes over time as shown at the right hand side of the figure.

The remainder of this paper introduces how the π-calculus can be used to represent data and processes required for orchestrations as well as additional ingredients to interactions such as correlations required for choreographies. It is supplemented by a running example and a section presenting application areas. We start by extending the motivation and introduce the π-calculus.

2 Motivation and Related Work

Figure 2 contains a choreography in BPMN [BPM04] notation. The choreography consists of the purchase process of a *Customer*. It dynamically includes a *Bank* found by a *Broker* if the purchase is above a threshold value. The *Customer's* pool shows the purchase process. First a data-based decision is made between the upper and lower part of the process. If the price of the purchase is lower then a given value, the purchase is made directly. If the price is above, a *Broker* is contacted to find a *Bank* with the lowest interests regarding the

purchase. Thereafter, the credit is requested at the *Bank*. In the considered scenario, the *Bank* can send two possible answers, either accept or reject the credit. This is evaluated in the *Customer's* process using a deferred choice. Based on the decision of the *Bank*, the purchase is made or rejected.

While the example choreography looks like a static interaction between *Customer*, *Broker*, and *Bank*, it is indeed an agile one. Only the *Broker* is known at design time, so it is directly coded into the *Find Bank* activity. However, the credit request relies on the answer of the *Broker's* lookup. Thus, the *Customer's* process dynamically binds to a certain *Bank* only known at runtime. The example contains typical ingredients of a SOA: internal processes with data (orchestrations) as well as static and dynamic interactions (choreographies).

The example can be discussed using approaches ranging from Petri nets to process algebra as well as existing standards. Petri nets have a long tradition for formalizing business processes [vAvH02]. They have also been extended to support distribution, e.g. Weske et al. in [vAW01]. Recent work investigated the representation of services and compatibility, e.g. Martens using *usability* in [Mar03], refined by Massuthe et al. using *operating guidelines* in [MRS05]. However, regarding distributed business processes, Petri net based approaches have two major drawbacks. First, they do not support more advanced routing pattern required for real-world business processes [vAtH03]. Second, they do not support interaction patterns that require dynamic binding [BDtH05]. Regarding the given example, only simple routing patterns are contained that cause no problems. However, the dynamic binding would generate infinite Petri nets if we assume an unknown number of possible participants. Extensions for special cases are possible but have a low general adequacy. Process algebra based approaches often neglect mobility aspects [BS05], but even if they consider it, there exist no investigations on full adequacy regarding advanced routing patterns. Thus, there exist no process algebra based approach until now that has been scientifically investigated regarding distributed business processes. Our motivation on investigating the π-calculus can be found in [Puh06].

3 The π-calculus

The π-calculus is based on a labeled transition system given by $(P, T, \xrightarrow{t})$, where P represents the set of states, i.e. all possible process descriptions, T is a set of transition labels, and $\xrightarrow{t}$ is a transition relation for each $t \in T$. To get started with the π-calculus, knowledge about process structures is sufficient, while the semantics can be used informally. Inside π-calculus processes *names* are used to represent links or pointers. Processes are denoted by uppercase letters and names by lowercase letters. The processes (i.e. states) of the π-calculus are given by:

$$\begin{aligned} P &::= M \mid P|P' \mid \mathbf{v}zP \mid !P \mid P(y_1, \cdots, y_n) \\ M &::= \mathbf{0} \mid \pi.P \mid M + M' \\ \pi &::= \overline{x}\langle\tilde{y}\rangle \mid x(\tilde{z}) \mid \tau \mid [x = y]\pi \ . \end{aligned} \tag{1}$$

The informal semantics is as follows: $P|P'$ is the concurrent execution of P and P', $\mathbf{v}zP$ is the restriction of the scope of the name z to P, $!P$ is an infinite number of copies of P, and $P(y_1, \cdots, y_n)$ represents recursion. $\mathbf{0}$ is inaction, a process that can do nothing, $M + M'$ is the exclusive choice between M and M'. The output prefix $\overline{x}\langle\tilde{y}\rangle.P$ sends a sequence of names $\tilde{y}$ via the co-name $\overline{x}$ and then continues as P. The input prefix $x(\tilde{z})$ receives a sequence of names via the name x and then continues as P with $\tilde{z}$ replaced by the received names (written as $\{^{n\tilde{a}me}/_{\tilde{z}}\}$). Matching input and output prefixes might communicate, leading to an interaction. The unobservable prefix $\tau.P$ expresses an internal action of the process, and the match prefix $[x = y]\pi.P$ behaves as $\pi.P$, if x is equal to y.

4 Unification

This section discusses, based on the example previously introduced, how the π-calculus can be used as a unified formal foundation for service oriented architectures. It starts by formalizing data, followed by processes, and is concluded by interactions.

4.1 Data

While the π-calculus is a process algebra and states are denoted as a syntactical description of a process given by the grammar from equation 1, it still allows for representing arbitrary data structures. Since pointers in the π-calculus are represented by names, names are used as accessors for data structures represented by processes. But how can static data like a boolean value be represented in the π-calculus? A straightforwards way is defining two restricted names for true and false, i.e. $(\mathbf{v}\top)$ for true and $(\mathbf{v}\bot)$ accordingly for false. These names are globally accessible by all processes inside the system, e.g.

$$\begin{aligned} SYSTEM &= (\mathbf{v}\top, \bot)(A \mid B) \\ A &= \tau.(\overline{ch}\langle\top\rangle.A + \overline{ch}\langle\bot\rangle.A) \\ B &= ch(x).([x = \top]\tau.B' + [x = \bot]\tau.\mathbf{0}) \ . \end{aligned}$$

The process $SYSTEM$ first creates two restricted names used for true and false and thereafter starts the processes A and B in parallel. However, only A can start execution immediately in the example, since B waits for a name x on ch. A does some internal calculation denoted by τ and thereafter sends either true or false via the name ch. In both cases, process A continues execution using recursion. Process B receives the name, where technically the placeholder x is replaced by the received name, i.e. either $\top$ or $\bot$. So, if A has sent $\top$, B evolves as follows:

$$ch(x).([x = \top]\tau.B' + [x = \bot]t.\mathbf{0}) \xrightarrow{ch(\top)} [\top = \top]\tau.B' + [\top = \bot]\tau.\mathbf{0} \ .$$

Since $\top \neq \bot$, only one possibility of the choice for B remains, making it deterministic (B executes τ and thereafter B'). If A had send $\bot$ instead, the second part of B would have

been executed. When looking at ch as a pointer, it clearly points to a process (A), that is able to return either true or false. Consequently, the *type* of the name ch can be said to be boolean, since it always points to a process representing boolean values in the example. By comparing two names of the type boolean, an AND process can be constructed:

$$\begin{aligned} AND = and(b1, b2, resp).b1(x).b2(y).([x = \top][y = \top]\overline{resp}\langle\top\rangle.AND + \\ [x = \bot]\overline{resp}\langle\bot\rangle.AND + \\ [y = \bot]\overline{resp}\langle\bot\rangle.AND) \ . \end{aligned}$$

The AND process is invoked via and with three parameters: Two names $b1$ and $b2$ representing pointers to booleans, and a third name used as response channel. First, AND fetches the actual values of the boolean pointers. Second, it returns true if both names, $b1$ and $b2$, are true and false otherwise. A possible extension for representing a byte instead of a boolean would be for A to return eight parameters of either true or false instead of one: $\overline{ch}\langle\bot, \bot, \top, \bot, \top, \bot, \top, \bot\rangle$ (representing decimal 42). One can now construct similar processes for adding, subtracting, or comparing bytes. Using these, other types can be implemented. Based on the observations made, we define a syntactical refinement to the π-calculus, namely typed names. A name is typed using common data types such as $\mathbf{v}i : Integer$. We further on introduce $[x == y], [x < y], [x > y]$, and $[x \neq y]$ for typed names similar to their obvious meaning. Technically, each of these construct is expanded to $(\mathbf{v}resp)\overline{proc_{type}}\langle x, y, resp\rangle$, where $proc_{type}$ calls a process for the corresponding operation and data type similar to AND.

After it has been shown how data can be represented in the π-calculus, it remains open how it can be made persistent, i.e. allowing for memory cells, stacks, queues, lists, and so on. A memory cell able to capture and allow modification of a single π-calculus name is denoted as:

$$\begin{aligned} CELL =& !(\mathbf{v}c)\overline{cell}\langle c\rangle.CELL_1(\bot) \\ CELL_1(n) =& \ \overline{c}\langle n\rangle.CELL_1(n) + c(x).CELL_1(x) \ . \end{aligned}$$

The process $CELL$ is replicated each time a fresh, restricted name c is read using $cell$. A new memory cell is initialized with the default name $\bot$ (false). The name c retrieved is then used as read and write accessor to the cells content. Thus, using c as an output prefix, the content of the memory cell can be changed, while using c as an input prefix, the content is read. Based on the memory cell, arbitrary data structures can be defined in the π-calculus.

4.2 Processes

After having introduced the core foundations for each information system, namely data, the representation of business processes in the π-calculus is investigated. Business processes are composed out of routing patterns that specify the control flow between activities. Examples are *Split*, *Join*, *Discriminator*, or patterns representing *Multiple Instances* of activities. A widely acknowledged catalogue of such patterns has been published by

van der Aalst et al. [vAtHKB00] as *Workflow Patterns*. In [PW05] we have shown how all these patterns can be represented in the π-calculus. In [OPW05] the pattern catalogue is extended by another pattern common in distributed business processes, called *Event-based Rerouting*. In general, each activity (including gateways etc.) of a business process is mapped to a π-calculus process, according to the following structure:

$$\{x_i\}_{i=1}^{m}.\{[a=b]\}_1^n.\tau.\{\overline{y_i}\}_{i=1}^{o}.\mathbf{0}\ . \tag{2}$$

Curly brackets are used to denote a static sequence of actions, the same holds for $\prod$ and $\sum$ for products and summations. Each activity consists of pre- and postconditions for routing the control flow using π-calculus names. The (abstracted) functional perspective of the activity is represented by the unobservable action τ. The precondition is split into two part: (1) $\{x_i\}_{i=1}^{m}$ denotes that the activity waits for m incoming names, and (2) $\{[a=b]\}_1^n$ denotes n additional guards that have to be true to execute the activity. The postcondition denotes the triggering of o outgoing names, i.e. $\{\overline{y_i}\}_{i=1}^{o}$. It is notable that the π-calculus representation replaces the type vs. instance view of business processes by a prototypical approach. Business processes are described formally in π-calculus and duplicated for execution.

By looking at the example again, the Workflow Patterns found in figure 2 are *Sequence* (BPMN sequence flow), *Exclusive Choice* and *Simple Merge* (BPMN exclusive OR gateways), as well as *Deferred Choice* (BPMN event-based gateway). A *Sequence* between two activities is represented by two different π-calculus processes:

$$A = \tau_A.\overline{b}.\mathbf{0} \qquad B = b.\tau_B.B'\ ,$$

where A signals the name b as a postcondition to B, that in turn can start execution. To make the formalization actually work, a third process, representing the composed system is required:

$$SYSTEM = (\mathbf{v}b)(A \mid B)\ .$$

Such a process is assumed for all further Workflow Patterns. An *Exclusive Choice* between two different activities B and C after an activity A is represented by:

$$A = \tau_A.(\overline{b}.\mathbf{0} + \overline{c}.\mathbf{0}) \qquad B = b.\tau_B.B' \qquad C = c.\tau_C.C'\ .$$

Note that the pattern given makes a non–deterministic choice between B or C. However, by using techniques such as $[x == y]$ as shown in the previous section, the choice can be made deterministic. The *Simple Merge* pattern waits for either one of the preceding activities, e.g. activity D waits for either B or C:

$$B = \tau_B.\overline{d_1}.\mathbf{0} \qquad C = \tau_C.\overline{d_2}.\mathbf{0} \qquad D = d_1.\tau_D.D' + d_2.\tau_D.D'\ .$$

A *Deferred Choice* pattern defers the decision until an external event occurs:

$$A = \tau_A.(b_{env}.\overline{b}.\mathbf{0} + c_{env}.\overline{c}.\mathbf{0}) \qquad B = b.\tau_B.B' \qquad C = c.\tau_C.C'\ .$$

After all patterns required for the example have been defined, the orchestrations from figure 2 can be formalized in π-calculus. Basically, each orchestration can be seen as a

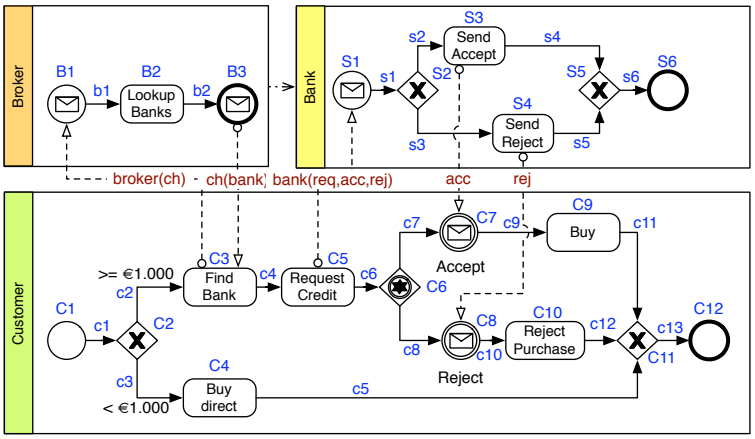


Figure 3: Example choreography annotated with identifiers.

strongly connected graph with exactly one initial and one final node. The nodes of the graph represent activities, events, or routing elements, while the edges represent dependencies between them. Each node has a semantics, e.g. an activity can be executed, an event can be consumed, or a routing decision can be made. Obviously, the semantics is given to the nodes by using the π-calculus pattern formalizations according to the following simplified algorithm:

- Assign all nodes of the graph a π-calculus process identifier.
- Assign all edges of the graph a unique π-calculus name.
- Define the π-calculus processes according to the π-calculus mapping of the Workflow Patterns found in [PW05, OPW05] as given by the type of the corresponding node. Each functional part of an activity is represented by the unobservable prefix τ since it is abstracted from concrete realizations.
- Define a global process that places all π-calculus processes representing nodes in parallel. □

An extended mapping approach from BPMN to π-calculus can be found in [PW06]. Regarding the example, we first have to annotate each node of the graphs representing the orchestrations with a π-calculus process identifier and each flow with a π-calculus name. The result is shown in figure 3. We also annotated the BPMN message flows, however this is not required until the complete choreography is defined in the next subsection. The global process (orchestration) of the broker is a quite simple:

$$BROKER = (\mathbf{v}b1, b2)(B1(b1) \mid B2(b1, b2) \mid B3(b2)) \ ,$$

with the following components representing the nodes:

$$B1(b1) = \tau_{B1}.\overline{b1}.\mathbf{0} \qquad B2(b1, b2) = b1.\tau_{B2}.\overline{b2}.\mathbf{0} \qquad B3(b2) = b2.\tau_{B3}.\mathbf{0} \ .$$

Initially, all nodes are placed in parallel in process $BROKER$. However, only component $B1$ can start immediately, since all other components require preconditions denoted by π-calculus names. The orchestration of the bank contains an exclusive decision that is not further specified. Due to space limitations, we only show the important π-processes:

$$S2(s1, s2, s3) = s1.\tau_{S2}.(\overline{s2}.\mathbf{0} + \overline{s3}.\mathbf{0})$$
$$S5(s4, s5, s6) = s4.\tau_{S5}.\overline{s6}.\mathbf{0} + s5.\tau_{S5}.\overline{s6}.\mathbf{0} \ .$$

The orchestration of the customer extends the representation of a decision node by taking data values into account, thus making the decision deterministic.[1] Again, we showcase the relevant process:

$$C2(c1, c2, c3) = c1(value).\tau_{C2}.([value > 999]\overline{c2}.\mathbf{0} + [value < 1000]\overline{c3}.\mathbf{0}) \ .$$

The π-calculus process $C2$ representing a data-based exclusive choice receives the $value$ to be evaluated from the preceding process. Based on the value, the decision is made. Note that $value$ is a pointer to a process representing a real number. Furthermore, $[value > 999]$ and $[value < 1000]$ are just placeholders for data-evaluation processes as introduced in the previous section. The representation of the event-based gateway will be given in the next subsection, since the decision is based on interactions between the customer and the bank.

4.3 Interaction

Agile choreographies, as contained in the example, are closely linked to the *Service Interaction Patterns* by Barros et al. [BDtH05]. These patterns describe possible behavior inside choreographies. Examples are *Send* and *Receive*, or *Dynamic Routing*, where the next interaction partner is determined by the current activity. In [DPW06] we have shown how the interaction patterns can be represented in the π-calculus. A synchronous service invocation is denoted as follows:

$$A = \overline{b}\langle msg\rangle.A' \qquad B = b(msg).B' \ ,$$

where A is the service requester and B is the service provider. The formalization leaves it open if A knows the link b since design time or acquired it during runtime. If it is defined as

$$S = (\mathbf{v}b)(A \mid B) \ ,$$

A and B share the link b since design time. Using link passing mobility in π-calculus, we can model a repository $R = \overline{lookup}\langle b\rangle.R$ that transmits the link at runtime:

$$S = (\mathbf{v}lookup)(lookup(b).A \mid ((\mathbf{v}b)B \mid R)) \ .$$

A common problem in the area of service oriented architectures is the description of correlations between service requesters and service providers. Usually, some kind of correlation

[1] This does not imply that the bank makes non-deterministic choices. However, the algorithm is not contained in the (abstract) orchestration.

identifier is placed inside each request and reply. The invoker as well as the provider have to take care to match all requests. In the π-calculus, the unique identifier of a request is also the channel used for reply from the service. By merging these two concepts, a clear representation of the correlations is straightforward. A new unique identifier is a π-calculus name, which is created with the $\mathbf{v}$ operator. A refined treatment of this topic can be found in [OPW05].

After having introduced the prerequisites, we are now ready to construct a complete formalization of the example. As already hinted in the previous subsection, the identifiers of the message flows between the different participants will be used therefore. First of all, the *Find Bank* activity of the *Customer* contacts a *Broker* known at design time. Therefore it uses a π-calculus name $broker$ that contains a restricted name ch used as a response channel:

$$C3(c2, c4, broker) = (\mathbf{v}ch)c2.\overline{broker}\langle ch\rangle.ch(bank).\tau_{C3}.\overline{c4}\langle bank\rangle.\mathbf{0} \ .$$

The π-calculus processes of the broker have to be changed accordingly:

$$\begin{aligned} B1(b1, broker) &= broker(ch).\tau_{B1}.\overline{b1}\langle ch\rangle.\mathbf{0} \\ B2(b1, b2, banklist) &= b1(ch).banklist(bank).\overline{b2}\langle ch, bank\rangle.\mathbf{0} \\ B3(b2) &= b2(ch, bank).\tau_{B3}.\overline{ch}\langle bank\rangle.\mathbf{0} \ . \end{aligned}$$

This time, the *Broker* can not immediately start working because $B1$ is waiting for an external request on the name $broker$. The name $banklist$ in $B2$ is a pointer to a priority list filled with *Banks* sorted by interests. By reading from this list a pointer to the *Bank* with the lowest interest is returned (i.e. the first item of the list). Finally, the pointer to the *Bank* is returned via ch in $B3$.

The *Customer's* deferred choice between *Accept* and *Reject* processing is implemented by $C6$, while the bank is first contacted at $C5$ (*Credit Request*):

$$\begin{aligned} C5(c4, c6) &= (\mathbf{v}req, acc, rej)c4(bank).\tau_{C5}.\overline{bank}\langle req, acc, rej\rangle.\overline{c6}\langle acc, rej\rangle.\mathbf{0} \\ C6(c6, c7, c8) &= c6(acc, rej).\tau_{C6}.(acc.\overline{c7}.\mathbf{0} + rej.\overline{c8}.\mathbf{0}) \ . \end{aligned}$$

The names acc and rej are used as environmental triggers for deciding the deferred choice. , whereas req represents the request. The complete choreography of the example is given by:

$$\begin{aligned} CHO(banklist) =& (\mathbf{v}broker)(BROKER(broker, banklist) \mid \\ & CUSTOMER(broker)) \ , \end{aligned}$$

with all the different *Banks* reachable over the $banklist$ pointer. Note that the actual management of the *Banks* such as adding or removing, is not contained inside the choreography. New *Banks* can register at runtime, whereas existing ones can withdraw or change their credit offers and conditions.

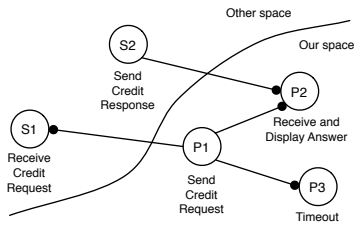


Figure 4: A distributed view of a service oriented architecture.

5 Application

The previous sections stated how the π-calculus can be used to model data, processes, and interactions found in service oriented architectures. Beside providing a uniform, unambiguous description of service oriented architectures, two further applications are of particular concern. The first one is execution and simulation, whereas the second one is reasoning.

Since the π-calculus is able to support data, orchestrations, as well as choreographies within one single algebra it can be seen as the least common denominator of service oriented architectures. Especially the direct support of dynamic binding by link passing mobility distinguishes the π-calculus from other approaches. By taking into account the results shown, execution environments for service oriented architectures should be based on the same concepts as the π-calculus. This minimal set of concepts already supports data, processes, and interactions as the core ingredients of service oriented architectures. Of course, the π-calculus is not applicable to business users or even most information technology experts. Thus, higher level constructs have to be introduced that can be broken down to π-calculus if required. For instance, a business process modeler uses the BPMN to model a business process that is then automatically mapped to π-calculus expressions for execution. Other results are more of a theoretical nature, for instance the representation of data types in the π-calculus. While a name can easily represent a pointer to some kind of data, such as an integer, the actual implementation has to be native for performance issues. Furthermore, the π-calculus describes a highly distributed representation of orchestrations and choreographies as shown in figure 4. In the figure, all activities of a choreography are shown as independent, distributed circles representing π-calculus processes. The processes use names to interact with each other, denoted by the lines between, where the dotted ends represent processes with the input prefix. While the activities of the right hand side belong inside *our* company, representing an orchestration, the left hand side activities belong to some other companies. Since the activities are all distributed and the link structure can be changed all the time, high flexibility is guaranteed. In a service oriented architecture, all activities are actually services reachable over *uniform resource locators* (URL) that can be deployed and interact as needed. The π-calculus provides a formal abstraction layer for such systems.

Another advantage of a formal representation of service oriented architectures are the

possibilities for reasoning. Especially in the context of open environments such as the Internet, where services are usually deployed and executed, a sound implementation is crucial. By using bisimulation techniques of the π-calculus [San93], orchestrations and choreographies can be checked before deployment and matchmaking of compatible services regarding the interaction behavior is possible. In [PW06] we have already shown how choreographies can be proved to be free of deadlocks and livelocks using a criterion called *Lazy Soundness*. An extension of this approach can be used to prove complete choreographies to be free of deadlocks and livelocks from the viewpoint of a certain orchestration. Thus, not only control flow dependencies are analyzed, but also interactions. Finally, the correctness of service implementations can be shown by proving the interaction equivalence between a service specification (i.e. an abstract orchestration) and a certain implementation.

6 Conclusion

In this paper we have shown how the π-calculus, an algebra for mobile systems, can be applied as a unified formal foundation for service oriented architectures. The unified formal representation of all key aspects of service oriented architectures — data, processes, and interactions — in one canonical minimal formal framework builds a foundation for further research and development. Yet recent standards like XLang (now superseded by BPEL4WS) [Mic01, BEA03] or WS-CDL [W3C04] claim to be based on the π-calculus. While they only took partial aspects of the π-calculus into consideration, recent research into service interaction patterns [BDtH05] identified the need for dynamic binding in service oriented environments. Until now, dynamic binding has been mostly neglected in scientific research. This paper highlighted the importance of the concepts found in the π-calculus for service oriented architectures.

References

[BDtH05] Alistair Barros, Marlon Dumas, and Arthur ter Hofstede. Service Interaction Patterns. In W.M.P. van der Aalst, Boualem Benatallah, and Fabio Casati, editors, *Proceedings of the 3rd International Conference on Business Process Management, volume 3649 of LNCS*, pages 302–318, Berlin, 2005. Springer-Verlag.

[BEA03] BEA Systems, IBM, Microsoft, SAP, Siebel Systems. *Business Process Execution Language for Web Services Version 1.1 (BPEL4WS)*, May 2003.

[BPM04] BPMI.org. *Business Process Modeling Notation*, 1.0 edition, May 2004.

[BS05] Lucas Bordeaux and Gwen Salaün. Using Process Algebra for Web Services: Early Results and Perspectives. In Ming-Chien Shan, Umeshwar Dayal, and Meichun Hsu, editors, *TES 2004, volume 3324 of LNCS*, pages 54–68, Berlin, 2005. Springer-Verlag.

[Bur00] Steve Burbeck. The Tao of E-Business Services. Available at: `http://www-128.ibm.com/developerworks/library/ws-tao/`, 2000.

[CCMS01] Erik Christensen, Francisco Curbera, Greg Meredith, and Weerawarana Sanjiva. *Web Service Description Language (WSDL) 1.1*. IBM, Microsoft, March 2001. W3C Note.

[DPW06] Gero Decker, Frank Puhlmann, and Mathias Weske. Formalizing Service Interactions. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *Proceedings of the 4th International Conference on Business Process Management (BPM 2006), volume 4102 of LNCS*, pages 414–419, Berlin, 2006. Springer Verlag.

[Mar03] Axel Martens. On Compatibility of Web Services. *Petri Net Newsletter*, 65:12–20, 2003.

[Mar05] Axel Martens. Analyzing Web Service based Business Processes. In Maura Cerioli, editor, *Proceedings of Intl. Conference on Fundamental Approaches to Software Engineering (FASE'05)*, volume 3442 of *Lecture Notes in Computer Science*. Springer-Verlag, April 2005.

[Mic01] Microsoft. *XLang Web Services for Business Process Design*, 2001.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, Part I/II. *Information and Computation*, 100:1–77, September 1992.

[MRS05] Peter Massuthe, Wolfgang Reisig, and Karsten Schmidt. An Operating Guideline Approach to the SOA. *Annals of Mathematics, Computing & Teleinformatics*, 1(3):35–43, 2005.

[OPW05] Hagen Overdick, Frank Puhlmann, and Mathias Weske. Towards a Formal Model for Agile Service Discovery and Integration. In Kunal Verma, Amit Sheth, Michal Zaremba, and Christoph Bussler, editors, *Proceedings of the International Workshop on Dynamic Web Processes (DWP 2005)*, IBM technical report RC23822, Amsterdam, December 2005.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1962.

[Puh06] Frank Puhlmann. Why do we actually need the Pi-Calculus for Business Process Management? In W. Abramowicz and H. Mayr, editors, *9th International Conference on Business Information Systems (BIS 2006), volume P-85 of LNI*, pages 77–89, Bonn, 2006. Gesellschaft für Informatik.

[PW05] Frank Puhlmann and Mathias Weske. Using the Pi-Calculus for Formalizing Workflow Patterns. In W.M.P. van der Aalst, Boualem Benatallah, and Fabio Casati, editors, *Proceedings of the 3rd International Conference on Business Process Management, volume 3649 of LNCS*, pages 153–168, Berlin, 2005. Springer-Verlag.

[PW06] Frank Puhlmann and Mathias Weske. Investigations on Soundness Regarding Lazy Activities. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *Proceedings of the 4th International Conference on Business Process Management (BPM 2006), volume 4102 of LNCS*, pages 145–160, Berlin, 2006. Springer Verlag.

[San93] Davide Sangiorgi. A Theory of Bisimulation for the Pi-Calculus. In *CONCUR '93: Proceedings of the 4th International Conference on Concurrency Theory*, pages 127–142, Berlin, 1993. Springer-Verlag.

[SBS04] Gwen Salaün, Lucas Bordeaux, and Marco Schaerf. Describing and Reasoning on Web Services using Process Algebra. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 43, Washington, DC, USA, 2004. IEEE Computer Society.

[vAtH03] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: Yet Another Workflow Language (Revised version. Technical Report FIT-TR-2003-04, Queensland University of Technology, Brisbane, 2003.

[vAtHKB00] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. Technical Report BETA Working Paper Series, WP 47, Eindhoven University of Technology, 2000.

[vAvH02] Wil van der Aalst and Kees van Hee. *Workflow Management*. MIT Press, 2002.

[vAW01] W. M. P. van der Aalst and M. Weske. The P2P Approach to Interorganizational Workflow. 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 LNCS*, pages 140–156, Berlin, 2001. Springer-Verlag.

[W3C04] W3C.org. *Web Service Choreography Description Language (WS-CDL)*, April 2004.

On the Suitability of WS-CDL for Choreography Modeling

Gero Decker,* Hagen Overdick
Hasso Plattner Institute, University of Potsdam, Germany
{gero.decker,hagen.overdick}@hpi.uni-potsdam.de

Johannes Maria Zaha
Queensland University of Technology, Australia
j.zaha@qut.edu.au

Abstract: The Web Service Choreography Description Language (WS-CDL) has been put forward as language for capturing sets of web service interactions and their control and data dependencies, seen from a global perspective. However the suitability of WS-CDL for this purpose has not been assessed in a systematic manner. This paper presents such an assessment by adopting a two-pronged approach. First, the paper studies the relation between WS-CDL and π-calculus, a well-known formalism for specifying communicating systems with dynamic topologies. Second, WS-CDL is assessed in terms of its support for two collections of patterns: the Workflow Patterns which capture recurrent control-flow dependencies in business processes, and the Service Interaction Patterns which capture recurrent compositions of interactions between services.

1 Introduction

With increasing maturity of implementation languages for Service-Oriented Architectures, languages for describing service interactions on a higher level of abstraction and in the early phases of the development lifecycle are emerging. WS-CDL allows one to define interactions, and to compose these interactions through control-flow dependencies in order to capture collaborative business processes. The Workflow Patterns [vdAtHKB03] are an established framework for evaluating the control-flow perspective of business process definition languages, while the Service Interaction Patterns [BDtH05] have been put forward as a benchmark for evaluating languages that support the definition of interactions and compositions thereof. Therefore, the combination of these two frameworks provides a basis for evaluating the suitability of WS-CDL for choreography definition. As a starting point, the relationship between WS-CDL and π-calculus is investigated. This allows for identifying the capabilities of WS-CDL to elevate the level of abstraction by providing higher-level constructs than π-calculus.

The paper is structured as follows. In Section 2 an introduction to π-calculus is given and the similarity between WS-CDL and π-calculus is shown. Sections 3 and 4 provide

*Research conducted while with SAP Research Centre Brisbane

solutions for the Workflow Patterns and the Service Interaction Patterns. Finally, Section 5 concludes.

2 WS-CDL and π-calculus

The π-calculus is a process algebra for mobile systems [MPW92]. In π-calculus, communication takes place between different π-processes. Names are a central concept in π-calculus. Links between processes as well as messages are names. This allows for link passing from one process to another. The scope of a name can be restricted to a set of processes but may be extruded as soon as the name is passed to other processes. The syntax for π-calculus looks as follows:

Listing 1: π-calculus syntax

$$\begin{aligned} P &::= M \mid P|P' \mid (\nu\ z)P \mid !P \\ M &::= 0 \mid \pi.P \mid M + M' \\ \pi &::= \overline{x}\langle y\rangle \mid x(y) \mid \tau \end{aligned}$$

Concurrent execution is denoted as $P|P'$, the restriction of the scope of z to P as $(\nu\ z)P$ and an infinite number of concurrent copies of P as $!P$ ("bang-operator"). Inaction of a process is denoted as 0. A non-deterministic choice between M and M' as $M + M'$ and sending y over x as $\overline{x}\langle y\rangle$. The prefix $x(y)$ receives a name over x and continues as P with y replaced by the received name. τ is the unobservable action. Communication between two processes can take place in the case of matching send- and receive-prefixes. It has been shown that all Service Interaction Patterns can be expressed using π-calculus ([DPW06]). As an example the Request with referral pattern, where a party A sends a request to party B indicating that a follow-up response should be sent to another party C, can be encoded as follows:

Listing 2: Request with referral

$$\begin{aligned} A &= (\nu\ a)\ \overline{b}\langle req, c, a\rangle\ .\ a(resp)\ .\ 0 \\ B &= b(req, x, y)\ .\ \overline{x}\langle resp, y\rangle\ .\ 0 \\ C &= c(msg, z)\ .\ \overline{z}\langle resp\rangle\ .\ 0 \end{aligned}$$

Party A creates a fresh name a which will be used as response channel later on. A passes the request together with a as well as a channel c to B. B receives the request, processes it and sends a response to the specified party. B does not need to know this third party in advance. In this example x is replaced with c so B uses this channel to pass on his response. C receives this message and sends his response back over the return channel a. Each π-process describes the publicly visible behavior of a participant ("interface process"). The combination of these behaviors determines the global interaction protocol. Each π-interaction in this setting represents an interaction between participants. In

contrast this example, π-interactions can also be used for control flow coordination within one participant. This strategy was followed in the formalization of all the Workflow Patterns ([PW05]) and some Service Interaction Patterns. Each activity within a process is represented by a π-process and coordination between these processes takes place via interactions. If a process participates in a collaboration with other processes, the interactions used for control flow coordination are not visible to the outside world.

Comparison. In the π-calculus example in Listing 2 we described "stitched" interface processes. Control flow is specified between communication actions which leads to an endpoint-centric view. The Web Services Choreography Interface standard (WSCI [A$^+$02]) follows this approach, too. In contrast to this, WS-CDL treats interactions between services as first-class citizens. Therefore, control flow is defined between interactions rather than communication actions.

Besides the direct support for channel passing in WS-CDL, especially the basic control flow constructs correspond to concepts in π-calculus. Some of them have different semantics though:

- `<sequence>` directly corresponds to a sequence in π-calculus (".").
- `<parallel>` corresponds to parallelism in π-calculus ("|"). However, the `<parallel>` construct of WS-CDL also implies merging after the parallel branches have completed. In π-calculus this is not the case. Here, we would have to introduce an additional interaction for synchronization purposes.
- *Non-blocking performs* (`<perform block="false">`) of choreographies also correspond to parallelism in π-calculus. No synchronization is needed after the performed choreography has completed. Using a non-blocking perform within an infinitely repeated work unit, the bang-operator "!" can be realized in WS-CDL.
- `<choice>` corresponds to a choice in π-calculus ("+"). Once again, merging the alternative branches is implied in WS-CDL which would have to be encoded differently in π-calculus.

Some WS-CDL constructs cannot be directly found in π-calculus:

- *(Non-blocking) guarded work units* (`<workunit guard=".." block="false">`) contain interactions that can only occur if the given condition evaluates to true. In π-calculus we could express this using a choice with two guarded alternatives where one alternative would be a τ_0-action. Furthermore, merging before follow-up activities would have to be encoded.
- *Repeated work units* (`<workunit repeat="..">`) are a convenient means to introduce repetitions into choreographies. In π-calculus either recursion or the bang-operator have to be used.

Like already mentioned in the previous section, π-interactions can be used to encode complex control flow. A direct support for this facility is missing in WS-CDL. However,

blocking work units can be used to realize a similar semantics in WS-CDL. Assume the following π-process:

$$(\nu\ h, i)\ (\tau_{AB}\ .\ (\bar{i}\ .\ 0 \mid \tau_{BC}\ .\ h\ .\ \tau_{BA}\ .\ 0) \mid \tau_{DB}\ .\ i\ .\ \tau_{BD}\ .\ \bar{h})$$

The τ-actions represent service interactions. The resulting global interaction protocol is: After interactions AB and DB, BD can happen. However, BC can already happen directly after AB without waiting for DB. Finally, BA can happen after BC and BD. Figure 1 illustrates this.

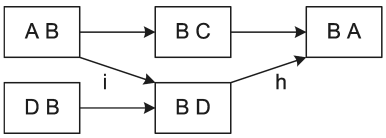


Figure 1: Sample choreography

We have introduced the fresh names h and i for realizing the desired control flow. In WS-CDL we can also introduce two variables h and i that cause the blocking of interaction BA and BD, respectively. For the rest of the control flow logic we use two sequences that run in parallel and assign a value to variables h and i at the appropriate places so that the blocked work unit are activated:

Listing 3: Sample choreography in WS-CDL

```
<parallel>
  <sequence>
    interactionAB
    <assign roleType="B"><copy>
      <source expression="true" />
      <target variable="cdl:getVariable('i')" />
    </copy></assign>
    interactionBC
    <workunit guard="cdl:getVariable('h')" block="true">
      interactionBA
    </workunit>
  </sequence>
  <sequence>
    interactionDB
    <workunit guard="cdl:getVariable('i')" block="true">
      interactionBD
    </workunit>
    <assign roleType="B"><copy>
      <source expression="true" />
      <target variable="cdl:getVariable('h')" />
    </copy></assign>
  </sequence>
</parallel>
```

π-calculus has a minimal number of language constructs in order to allow for elegant reasoning. In contrast to this, WS-CDL is more of an engineering medium to facilitate the design of service choreographies and therefore includes more high-level concepts. Unlike

π-calculus, WS-CDL is typed and has a high expressiveness for specifying data structures ("information types") and conditions such as repetition conditions and guard conditions. Role types, relationship types, participant types and channel types can be easily described. Identity tokens definitions are a convenient way to capture correlation.

Since WS-CDL is tightly coupled with web service technology and does not provide any graphical representations, we argue that WS-CDL should only be used in the latest stages of the choreography definition process. Like it is proposed to use the Business Process Modeling Notation (BPMN [bpm06]) prior to encoding process descriptions in BPEL, there should be a similar choreography modeling notation for early choreography design stages. UML2.0 Activity Diagrams have an extensible meta-model that could used for defining a choreography notation. However, to the best knowledge of the authors there is no publication introducing such an extension.

3 Workflow Pattern support

The Workflow Patterns were introduced by van der Aalst et al. in [vdAtHKB03]. They serve as a reference framework for assessing process modeling languages and workflow systems in terms of control flow expressiveness. WSCI as well as a number of other standards (e.g. BPEL, UML 2.0 Activity Diagrams, BPMN) have already been assessed ([vdADtHW02], [WvdADtH03], [RvdAtHW06], [WvdAD$^+$06]). These assessments serve as reference for identifying if direct, partial or no support for a particular pattern is given.

3.1 Basic control flow patterns

Sequence is directly supported through the structure type `<sequence>` in WS-CDL. As shown above, control flow dependencies between interactions could also be realized through blocking work units and variable assignments.

Corresponding ***Parallel Splits*** and ***Synchronizations*** can be expressed using `<parallel>` structures. Although only block structures can be captured, we follow the assessments for these two patterns in XLang ([WvdADtH03]) and BPML / WSCI ([vdADtHW02]) and therefore conclude that there is direct support for them. A workaround for encoding arbitrary *Synchronizations* is the usage of blocking work units and variables (cf. Listing 3). A *Parallel Split* without synchronization can alternatively be expressed using non-blocking `<perform>` structures.

The ***Exclusive Choice*** pattern appears as the structure type `<choice>` in WS-CDL. It specifies that only one activity within the structure is selected and all other activities are disabled. Different control flow behaviors apply depending on whether or not work units with guard conditions are contained in the structure. In the case where such work units are present the behavior of an *Exclusive Choice* applies. The first work unit that matches the guard condition is selected. `<choice>` also implements ***Simple Merge*** at the same time.

Thus direct support for both patterns. Arbitrary *Simple Merges* can again be implemented using blocking work units as a workaround.

3.2 Advanced Branching and Synchronization patterns

In the case of the ***Multiple Choice*** pattern a number of branches are chosen. This can be expressed using a `<parallel>` structure containing guarded work units. A matching ***Synchronizing Merge*** is also covered by this structure.

Multiple Merge: A point in a process where two or more branches reconverge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started for every activation of every incoming branch. Solution: Non-blocking `<perform>` structures can be used to implement this pattern. A sub-choreography is defined and instances are performed from within parallel activities. This style of implementation is similar to spawning off new process instances in BPML, which causes problems if the *Multi Merge* is placed within a cycle. In analogy to BPML, we opt for partial support for this pattern.

The ***Discriminator*** is a point in a process that waits for one of the incoming branches to complete before activating the subsequent activity. Solution: A blocking work unit can be used to emulate a *Discriminator*. Several activities running in parallel assign a value to a synchronization variable. As soon as this happens the blocking work unit can start. Since there is no direct language construct for the *Discriminator* and implementations always involve bookkeeping through variables, there is no support for this pattern.

3.3 Structural patterns

Arbitrary Cycles is a point in a process where one or more activities can be done repeatedly. *Arbitrary Cycles* are not supported in WS-CDL. Cycles with one entry and one exit point are implemented using repeated work units.

Implicit Termination: A given sub-process should be terminated when there is nothing else to be done. This pattern covers how to describe the point of termination of an instance of the model. In the case of WS-CDL there is direct support for this pattern: Sub-choreography instances can be activated using non-blocking performs and there is thus no defined point in the choreography where termination of the choreography instance happens.

3.4 Patterns involving Multiple Instances

These patterns describe scenarios where multiple instances of an activity can be created in the context of a single case. If the number of instances is known at design-time (***MI***

with a priori design time knowledge) the activities could be replicated and placed in a <parallel> structure. These structures also indicate synchronization after completion of the activities. Alternatively, a sub-choreography can be defined and activated several times using <perform> actions. These actions can either be blocking or non-blocking while in the latter case no synchronization takes place (***MI without synchronization***).

The case where the number of instances is not known at design-time (***MI with a priori runtime knowledge*** and ***MI with no a priori runtime knowledge***) cannot directly be expressed in WS-CDL. As a workaround, blocking work units and variables can be used: A sub-choreography is defined and performed several times within a repeated work unit. If the <perform> action is non-blocking, the sub-choreography instances are activated in parallel. In the case of no a priori knowledge the repeated work unit is also guarded so instances can be activated at a later point in time. A counter is used to record how many instances have already completed. A blocking work unit is activated as soon as the counter has reached a certain value.

Listing 4: Workaround for MI with a priori runtime knowledge

```
<choreography name="chor1">
  <sequence>
    activity
    <assign><copy><!-- increase counter -->
        <source expression="cdl:getVariable('counter') + 1" />
        <target variable="cdl:getVariable('counter')" />
    </copy></assign>
  <sequence>
</choreography>..
  <sequence>
    <assign><copy><!-- initialize counter -->
      <source expression="0" />
      <target variable="cdl:getVariable('counter')" />
    </copy></assign>
    <workunit .. repeat=".."><!-- spawn off multiple instances -->
      <perform choreographyName="chor1" block="false" />
    </workunit>
    <workunit guard="cdl:getVariable('counter')==.." block="true" ..>
      ...
    </workunit>
 </sequence>
```

3.5 State-based Patterns

A ***Deferred Choice*** is a point in the process where one of several branches is chosen. In contrast to the XOR-split, the choice is not made explicitly (e.g. based on data or a decision) but several alternatives are offered to the environment. This means that once the environment activates one of the branches the other alternative branches are withdrawn. It is important to note that the choice is delayed until the processing in one of the alternative branches is actually started. Solution: <choice> structures implement both *Exclusive*

Choice and *Deferred Choice*. If no guarded work units are contained in a choice structure the decision criteria is hidden and *Deferred Choice* behavior applies.

In the case of ***Interleaved Parallel Routing*** a set of activities is executed in an arbitrary order: Each activity in the set is executed, the order is decided at run-time, and no two activities are executed at the same moment (i.e. no two activities are active for the same workflow instance at the same time). There is no support for Interleaved Parallel Routing in WS-CDL. A possible workaround could be to introduce a variable representing a mutex. Blocking work units are used the guard condition of which evaluates to true if the mutex is available (i.e. the variable a certain value). Then as a first activity within each work unit the mutex is made unavailable (i.e. another value is assigned). However, this is only possible if the first activity is performed before another work unit is activated. This might depend on the underlying implementation.

Milestone: The enabling of an activity depends on the case being in a specified state, i.e. the activity is only enabled if a certain milestone has been reached which did not expire yet. Solution: In WS-CDL state can be represented using variables ("state capturing variables"). Work units can be activated if a certain guard condition evaluates to true. Milestones are then expressed by non-blocking work units.

3.6 Cancellation patterns

In the case of ***Cancel Activity*** an enabled activity is disabled, i.e. a thread waiting for the execution of an activity is removed. Solution: In WS-CDL exceptions can be caused which lead to disabling activities within the scope of the nearest exception handler. Thus direct support.

Cancel Case is directly supported, too. A case, i.e. workflow instance, is removed completely (i.e., even if parts of the process are instantiated multiple times, all descendants are removed). This can be achieved through exceptions that are handled in the root choreography.

4 Service Interaction Pattern support

The Service Interactions Patterns were introduced by Barros et al. in [BDtH05]. They present common interaction scenarios between two or more parties and can be used to assess choreography languages. Although [BDtH05] also contains hints about how different languages implement individual patterns no complete assessment of a standard has been carried out so far using this set of patterns.

4.1 Single-transmission bilateral interaction patterns

All three patterns ***Send***, ***Receive*** and ***Send/receive*** are directly supported in WS-CDL. The structure type `<interaction>` allows to define message exchanges between two services. The `action` attribute of a channel type specifies whether the message exchange is of type `request-only`, `response-only` or `request-response`. Binding a particular participant for an interaction is realized through assigning channel instances to variables. At any point in a choreography a new instance can be assigned. Therefore, design-time and runtime binding and even runtime re-binding of participants can be expressed.

4.2 Single-transmission multilateral interaction patterns

The ***Racing incoming messages*** pattern is similar to the Workflow Pattern *Deferred Choice*: A party expects to receive one among a set of messages. It can be expressed using a `<choice>` containing interactions with the same recipient.

In the case of the ***One-to-many send*** pattern a party sends messages to several parties. If the number of recipients is known at design-time, interactions can be placed in a `<parallel>` structure. This can represent both design-time and runtime binding of recipients of a specified role. However, if the number of recipients is not known at design-time we have the same problem like in the *MI with a priori runtime knowledge* pattern. As a workaround the interactions could be serialized in a repeated work unit. Or an interaction is placed in a sub-choreography which is activated several times using non-blocking `<perform>` actions. A variable is used for bookkeeping how many sends have already completed. Since WS-CDL only directly supports the case where the number of recipients is known at design-time, we opt for partial support for this pattern.

The ***One-from-many receive*** pattern describes that a party receives a number of logically related messages that arise from autonomous events occurring at different parties. The arrival of messages needs to be timely so that they can be correlated as a single logical request. Solution: This can be expressed using a repeated work unit containing a single interaction.

The ***One-to-many send/receive*** is very similar to *One-to-many send*: A party sends a request to several other parties. Responses are expected within a given timeframe. The interaction may complete successfully or not depending on the set of responses gathered. Solution: The timeframe aspect is directly supported through the `<timeout>` structure in WS-CDL. Successful vs. unsuccessful completion is directly supported through exception mechanisms. However, we have the same problem with an unknown number of participants at design-time like it was the case for *One-to-many-send*. Hence, there is only partial support for this interaction pattern, too.

4.3 Multi-transmission interaction patterns

In the case of ***Multi-responses*** a party X sends a request to another party Y. Subsequently, X receives any number of responses from Y until no further responses are required. Solution: This pattern is directly supported through a repeated work unit containing the response interaction.

Contingent requests: A party X makes a request to another party Y. If X does not receive a response within a certain timeframe, X alternatively sends a request to another party Z, and so on. Responses from previous requests might be still considered or discarded. Solution: The limited timeframe can be specified through the `<timeout>` structure. A work unit is activated several times until a response arrives before the timeout occurs or the list of potential recipients has been reached. For every iteration a new channel instance is assigned to the channel variable. The selection of the next participant would be hidden. However, this only covers the cases where responses of previous requests are discarded. The case where other responses are still considered requires a parallel execution of the request-response-interactions. Blocking work units could be used to ensure that interactions are only activated a certain time after the previous request has been activated. Nevertheless, this only works for the case where the number of recipients is known at design-time. Since the discard-case is directly supported we still conclude that there is partial support for the pattern.

In the case of ***Atomic multicast notification*** a party sends notifications to several parties such that a certain number of parties are required to accept the notification within a certain timeframe. For example, all parties or just one party are required to accept the notification. In general, the constraint for successful notification applies over a range between a minimum and maximum number. There is no direct support for this pattern in WS-CDL. As a workaround we could resort to blocking work units and variables: A parallel structure contains several perform actions as well as two blocking work units. In the sequences the actual interactions take place. Depending on the outcome of the interaction a counter is increased. As soon as the counter reaches a certain value or a global timeout occurs, one of the blocking work units is activated.

4.4 Routing patterns

Request with referral: Party A sends a request to party B indicating that any follow-up response should be sent to a number of other parties (P1, P2, ..., Pn) depending on the evaluation of certain conditions. Solution: WS-CDL directly supports channel passing. An `<exchange>` structure within an `<interaction>` structure can contain a specification of the corresponding `channelType`. Although the follow-up responses might need to be sent to a number of participants that is only known at runtime (which would not be supported by WS-CDL), we argue that the notion of channel passing is at the very core of the pattern. So we conclude that there is direct support.

```
<channelType name="Channel1">..</channelType>
```

```
<interaction>
  <exchange channelType="Channel1">..</exchange>..
</interaction>
```

Relayed request: Party A makes a request to party B which delegates the request to other parties (P1, ..., Pn). Parties P1, ..., Pn then continue interactions with party A while party B observes a "view" of the interactions including faults. Solution: All responses are encoded in one-way interactions. Using a `<parallel>` structure responses are sent to A and to B.

Dynamic routing: A request is required to be routed to several parties based on a routing condition. The routing order is flexible and more than one party can be activated to receive a request. When the parties that were issued the request have completed, the next set of parties are passed the request. Routing can be subject to dynamic conditions based on data contained in the original request or obtained in one of the "intermediate steps". Since the pattern description is very unprecise it is hard to judge whether or not this pattern is supported. Dynamic conditions in the sense that a participant can overwrite e.g. repetition or guard conditions at runtime is not supported. Static routing orders can easily be expressed using `<sequence>` and `<parallel>` structures. Dynamic routing orders in the sense that a participant can delete interactions or insert new interactions into the choreography at runtime is not supported. We skip this pattern in the assessment.

5 Conclusion

This paper has discussed the relationship between WS-CDL and π-calculus. It turns out that π-calculus and WS-CDL share some elements, but that a number of high-level constructs are provided in WS-CDL that result in more direct pattern support.

Table 1 summarizes which Workflow Patterns and Service Interaction Patterns are supported in WS-CDL. In analogy to the mentioned assessments of other process modeling languages we assign a "+" for direct support of a pattern, "+/–" for partial support and "–" for lack of support. The table shows that WS-CDL supports more patterns than its predecessor WSCI. However, we propose introducing a construct similar to `<forEach>` in BPEL 2.0. This would lead to direct support of *Multiple Instances with a priori runtime knowledge* as well as the three Service Interaction Patterns that are currently only partially supported. At the moment variable assignments and blocking work units have to be resorted to as workarounds. This is problematic since mappings of blocking work units to implementation languages such as BPEL still remain unclear.

We stated that WS-CDL is not suited for early choreography design stages and we called for a standardized graphical notation, e.g. an extension to UML2.0 Activity Diagrams. WS-CDL could then be used as intermediary language introducing web-service-specific definitions.

Acknowledgments. The second author is funded in part by SAP. Many thanks to Marlon Dumas for his valuable feedback.

Workflow Patterns	WS-CDL	WSCI	BPEL
1. Sequence	+	+	+
2. Parallel Split	+	+	+
3. Synchronization	+	+	+
4. Exclusive Choice	+	+	+
5. Simple Merge	+	+	+
6. Multiple Choice	+	–	+
7. Synchronizing Merge	+	–	+
8. Multiple Merge	+/–	+/–	–
9. Discriminator	–	–	–
10. Arbitrary Cycles	–	–	–
11. Implicit Termination	+	+	+
12. MI without synchronization	+	+	+
13. MI with a priori design time knowledge	+	+	+
14. MI with a priori runtime knowledge	–	–	–
15. MI with no a priori runtime knowledge	–	–	–
16. Deferred Choice	+	+	+
17. Interleaved Parallel Routing	–	–	+/–
18. Milestone	+	–	–
19. Cancel Activity	+	+	+
20. Cancel Case	+	+	+
Service Interaction Patterns	WS-CDL		
1. Send	+		
2. Receive	+		
3. Send/receive	+		
4. Racing incoming messages	+		
5. One-to-many send	+/–		
6. One-from-many receive	+		
7. One-to-many send/receive	+/–		
8. Multi-responses	+		
9. Contingent requests	+/–		
10. Atomic multicast notification	–		
11. Request with referral	+		
12. Relayed request	+		

Table 1: Pattern support in WS-CDL

References

[A+02] Assaf Arkin et al. Web Service Choreography Interface (WSCI) 1.0. Technical report, Aug 2002. `http://www.w3.org/TR/2002/NOTE-wsci-20020808`.

[BDtH05] Alistair Barros, Marlon Dumas, and Arthur ter Hofstede. Service Interaction Patterns. In *Proceedings 3rd International Conference on Business Process Management (BPM 2005)*, pages 302–318, Nancy, France, 2005. Springer Verlag.

[bpm06] Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification. Technical report, Object Management Group (OMG), February 2006. http://www.bpmn.org/.

[DPW06] Gero Decker, Frank Puhlmann, and Mathias Weske. Formalizing Service Interactions. In *Proceedings 4th International Conference on Business Process Management (BPM 2006)*, Vienna, Austria, Sept 2006. Springer LNCS.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–40, 1992.

[PW05] Frank Puhlmann and Mathias Weske. Using the pi-Calculus for Formalizing Workflow Patterns. In *Proceedings 3rd International Conference on Business Process Management (BPM 2005)*, pages 153–168, Nancy, France, 2005. Springer Verlag.

[RvdAtHW06] N. Russell, Wil M.P. van der Aalst, Arthur ter Hofstede, and Petia Wohed. On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. In *Proceedings 3rd Asia-Pacific Conference on Conceptual Modelling (APCCM 2006)*, volume 53 of *CRPIT*, pages 95–104, Hobart, Australia, 2006.

[vdADtHW02] Wil M.P. van der Aalst, Marlon Dumas, Arthur H.M. ter Hofstede, and Petia Wohed. Pattern-Based Analysis of BPML (and WSCI). QUT Technical report FIT-TR-2002-05, Queensland University of Technology, Brisbane, Australia, 2002.

[vdAtHKB03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[WvdAD$^+$06] Petia Wohed, Wil M.P. van der Aalst, Marlon Dumas, Arthur ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. In *Proceedings 4th International Conference on Business Process Management (BPM 2006)*, LNCS, Vienna, Austria, 2006. Springer Verlag.

[WvdADtH03] Petia Wohed, Wil M.P. van der Aalst, Marlon Dumas, and Arthur ter Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In *Proceedings 22nd International Conference on Conceptual Modeling (ER 2003)*, volume 2813 of *LNCS*, pages 200–215. Springer Verlag, 2003.

Modellierung von Zugriffsrichtlinien für offene Systeme

Wolfgang Dobmeier und Günther Pernul

Lehrstuhl für Wirtschaftsinformatik I
Universität Regensburg
D-93040 Regensburg
{wolfgang.dobmeier, guenther.pernul}@wiwi.uni-regensburg.de

Abstract: Für offene Systeme im Internet wie Web Services ist die Sicherheit der angebotenen Dienste von entscheidender Bedeutung. Ein wichtiger Baustein dabei ist die Umsetzung der Zugriffskontrolle. Dazu sind besondere Zugriffskontrollmodelle nötig, die über traditionelle Ansätze hinausgehen. Bisher fehlte es jedoch an einer Möglichkeit, Zugriffsrichtlinien für solche Modelle in leicht verständlicher Form spezifizieren zu können. Der Beitrag stellt neben einem passenden Zugriffskontrollmodell eine pragmatische Methode zur Modellierung von Zugriffsrichtlinien in Form eines UML-Profils vor. Die Methode wird an einem Beispielszenario illustriert.

1 Einleitung

Offene Systeme gewinnen in der modernen Zeit an Bedeutung. Beispiele für solche Systeme können allgemeine Anwendungen des eCommerce oder eGovernment-Portale sein. Eine Möglichkeit, offene Systeme technisch umzusetzen, sind Web Services. So könnte ein Unternehmen Dienste für jeden Benutzer offen nach außen hin anbieten, wie es z.B. Amazon[1] mit einem umfangreichen Spektrum, von der Abfrage von Preisen bis hin zur Abwicklung von privaten Buchangeboten, demonstriert.

Das bedeutet, dass diese Dienste auch geschäftskritische Bedeutung haben. Daher muss auf die Sicherheit dieser Dienste besonderes Augenmerk gelegt werden. Offene Systeme weisen zudem einige Charakteristika auf, die in herkömmlichen Systemen nicht anzutreffen sind. So werden z.B. anfragende Benutzer oftmals dem System nicht a priori bekannt sein, außerdem kann die Anzahl von Ressourcen und Benutzern sehr groß sein. Auf der einen Seite sollen also Ressourcen all denjenigen, die dafür berechtigt sind, zugänglich gemacht werden, auf der anderen Seite zugleich vor einem Zugriff durch Unberechtigte geschützt werden. Diese Aufgabe kommt der Zugriffskontrolle zu. Durch die Definition von Zugriffsrichtlinien (Policies) kann festgelegt werden, welche Subjekte auf welchen Objekten welche Operationen durchführen können. Dies war bereits in der Vergangenheit eine fehleranfällige und mühsame Tätigkeit.

Aus den o.g. Gründen muß die Zugriffskontrolle jedoch in offenen Systemen leistungsfähiger als in herkömmlichen Systemen gestaltet werden. Dies manifestiert sich zum Einen in

[1] `http://solutions.amazonwebservices.com`

der Verwendung neuartiger Modelle zur Zugriffskontrolle [DDCdVS05], die auf den Attributen von Subjekten und Objekten basieren (siehe Abschnitt 2). Zum Anderen hat sich unter dem Gesichtspunkt der Anwendungsarchitektur dabei eine logische Trennung der Bereiche Spezifikation der Zugriffsrichtlinien, Fällen der Zugriffskontrollentscheidung und Durchsetzen der Entscheidung herausgebildet. So definiert der offene Standard XACML [OAS05] neben einem XML-Dialekt zum Beschreiben von Zugriffsrichtlinien eine generische Architektur, die u.a. zwischen einem Policy Administration Point (PAP), Policy Decision Point (PDP) und Policy Enforcement Point (PEP) unterscheidet. Andere Autoren in der Literatur wie z.B. [YT05] folgen dieser Trennung in ähnlicher Weise.

Durch diese Separierung ist es also möglich, die Zugriffsrichtlinien getrennt von der Zugriffskontrolllogik zu verwalten, so dass dazu auch spezielle Werkzeuge mit eigenen Spezifikationsmöglichkeiten verwendet werden können. Jedoch mangelt es bisher daran. Durch die höhere Flexibilität (und Komplexität) attributbasierter Zugriffskontrollmodelle ist die Textform zur Dokumentation der Zugriffsrichtlinien nur eingeschränkt geeignet. So ist die XML-basierte Notation von XACML für den menschlichen Leser schwer zugänglich. Es gibt zwar vereinzelt Editoren für XACML (z.B. der UMU-XACML-Editor[2]), diese können jedoch nicht intuitiv bedient werden. Auch andere Möglichkeiten der Spezifikation von Richtlinien, wie z.B. graphbasierte [KMPP05] oder logikbasierte Ansätze [ZRG04], sind für ungeschulte Benutzer eher schwer verständlich. Da eine einfache und damit verständliche Definition der Zugriffsrichtlinien aber essentiell für die Sicherheit eines Systems ist, folgern wir, dass eine übersichtliche und gleichzeitig aber leistungsfähige Methode zur Richtlinien-Spezifikation notwendig ist. Im vorliegenden Beitrag präsentieren wir auf Basis bestimmter Anforderungen einen pragmatischen Ansatz zur grafischen Spezifikation von Zugriffsrichtlinien für offene Systeme. Er ist eine Weiterentwicklung des in [PDMP05] nur rudimentär dargelegten Vorgehens.

Im weiteren Verlauf des Beitrags wird in Abschnitt 2 ein auf die Anforderungen von offenen Systemen zugeschnittenes Zugriffskontrollmodell behandelt. Abschnitt 3 führt ein UML-Profil ein, mit dem Richtlinien für das vorgestellte Modell entworfen werden können; dies wird an einem Beispielszenario verdeutlicht. Der 4. Abschnitt geht auf verwandte Arbeiten ein. In Abschnitt 5 wird abschließend ein Ausblick auf zukünftige Entwicklungen gegeben.

2 Zugriffskontrolle für offene Systeme

Wie bereits in Abschnitt 1 erwähnt, weist das Problem der Zugriffskontrolle in offenen Systemen einige Besonderheiten auf. Traditionelle Zugriffskontrollmodelle wie die rollenbasierte Zugriffskontrolle (RBAC) [FSG+01], die sich am Aufbau von Organisationen orientiert, sind ungeeignet für den Einsatz an dieser Stelle. Gründe liegen u.a. darin, dass traditionelle Modelle identitätsbasiert sind, während in offenen Systemen die Identität von Subjekten zugunsten aller Eigenschaften der Subjekte in den Hintergrund tritt. Außerdem erschwert die potenziell große Zahl von Subjekten und Objekten das Management der

[2] `http://xacml.dif.um.es`

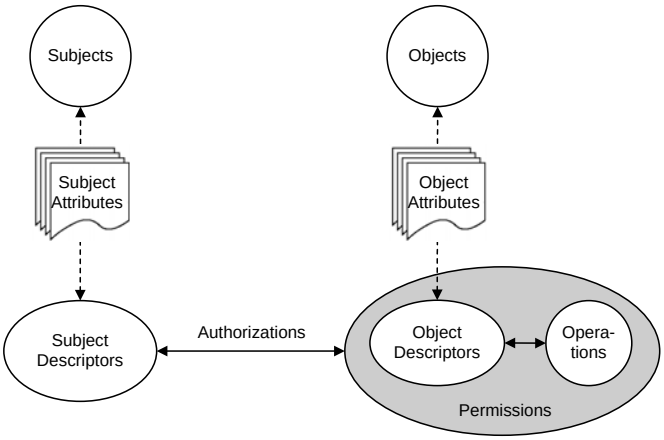


Abbildung 1: Attributbasierte Zugriffskontrolle

Zugriffsrichtlinien [PDMP05].

Im Folgenden stellen wir kurz ein Modell zur attributbasierten Zugriffskontrolle (ABAC) vor, das den genannten Besonderheiten gerecht wird. Eine erste, ausführlichere Beschreibung des Modells ist in [PFMP04] enthalten.

Die Grundidee attributbasierter Zugriffskontrolle ist in Abbildung 1 zu sehen. Sie besteht darin, Zugriffsrechte zwischen den Subjekten und Objekten nicht statisch zu definieren, sondern ihre Eigenschaften oder Attribute dynamisch als Grundlage der Autorisierung zu nutzen. Die Attribute von Subjekten können eher statischer Natur sein, wie z.B. die Position eines Benutzers in einem Unternehmen; sie können jedoch auch dynamischer Natur sein, wie z.B. der aktuelle Aufenthaltsort, das Alter oder auch ein erworbenes Abonnement für eine digitale Bibliothek. Auf Seiten der Objekte werden Metadaten verwendet wie z.B. der Typ oder das Thema einer Ressource. Subjekte und Objekte werden jeweils von einer Menge an Attributen sowie den zugehörigen Werten repräsentiert. Permissions, d.h. Privilegien, bestehen aus der Kombination eines sog. Objektdeskriptors und einer Operation (z.B. Lesen oder Schreiben), die auf den durch den Objektdeskriptor bezeichneten Objekten durchgeführt werden soll. Deskriptoren bestehen allgemein aus einer Menge von Attributen, die jeweils Bedingungen unterworfen werden wie z.B. ’Alter > 17’ oder ’Abonnent = ja’. Eine Rechtezuweisung (Authorization) wird dann zwischen einem Subjektdeskriptor und einem Privileg vorgenommen. Mit Hilfe von Deskriptoren ist es möglich, die im System existierenden Objekte bzw. die anfragenden Subjekte dynamisch den für sie passenden Rechtezuweisungen zuzuordnen und so eine manuelle Rechtevergabe entbehrlich zu machen.

Das ABAC-Modell ist mächtig genug, um die traditionellen Zugriffskontrollmodelle wie benutzerbestimmbare Zugriffskontrolle (DAC), systembestimmte Zugriffskontrolle (MAC) und RBAC abbilden zu können. Genaueres findet sich in [PDMP05].

3 Grafische Modellierung von Zugriffsrichtlinien

In diesem Abschnitt wird eine grafische Methode zum Entwurf von Zugriffsrichtlinien für attributbasierte Modelle entwickelt. Dazu werden in einem ersten Schritt einige Anforderungen, die eine solche Methode aus unserer Sicht erfüllen sollte, aufgestellt:

- Es sollte eine etablierte grafische Notation verwendet werden, so dass die Spezifikationsmethode in die in der Industrie gängigen Entwicklungsprozesse integriert werden kann.
- Ausreichende Werkzeugunterstützung in Form dedizierter Tools oder eine Integration mit verbreiteten Werkzeugen sollte möglich sein; auch im Hinblick auf die spätere automatisierte Transformation der grafischen Spezifikation in eine Repräsentation, die von einem PDP verwendet werden kann.
- Die der grafischen Notation zugrunde liegende Modellierungsprache sollte leichte Erweiterbarkeit aufweisen, so dass keine umfangreiche Neuentwicklung erfolgen muss und die vorhandenen Modellierungswerkzeuge weiter benutzt werden können.

Die Unified Modeling Language (UML) scheint geeignet, die beschriebenen Anforderungen zu erfüllen. Als de-facto Standard für die Modellierung von Softwaresystemen ist sie weitverbreitet und wohlverstanden. Zudem bietet sie einige Ansatzpunkte unterschiedlicher Mächtigkeit zur Erweiterung mit eigenen Sprachkonstrukten. Neben der Möglichkeit, ein vollständig neues Metamodell zu definieren, kann auch das UML-Metamodell unkontrolliert bzw. kontrolliert erweitert werden [HKKR05, S. 334]. Um die bestehende Semantik zu erhalten, verwenden wir die Variante der kontrollierten Erweiterung in Form eines UML-Profils. Da diese Variante auch von den meisten UML-Modellierungstools implementiert werden kann, ist damit die Grundlage für eine umfassende Werkzeugunterstützung gelegt.

3.1 ABAC-Richtlinien mit UML

Dieser Abschnitt stellt zunächst ein konzeptuelles Modell der ABAC-Richtlinien vor. Danach wird eine dedizierte UML-Notation für ABAC-Richtlinien in Form eines Profils erläutert.

Konzeptuelles Muster. Abbildung 2 zeigt ein Muster zum grundlegenden Verständnis der Elemente der attributbasierten Richtlinienspezikation. Es basiert in leicht abgewandelter Form auf den Entwicklungen in [PDMP05], ist jedoch für die Modellierung von Richtlinien angepaßt. So erfolgt im Muster z.B. keine Modellierung von konkreten Benutzern und Ressourcen des Anwendungssystems, da das ABAC-Modell ja gerade von einzelnen Identitäten abstrahiert.

Die Beschreibung der Klassen wurde großenteils bereits analog in Abschnitt 2 vorgenommen.

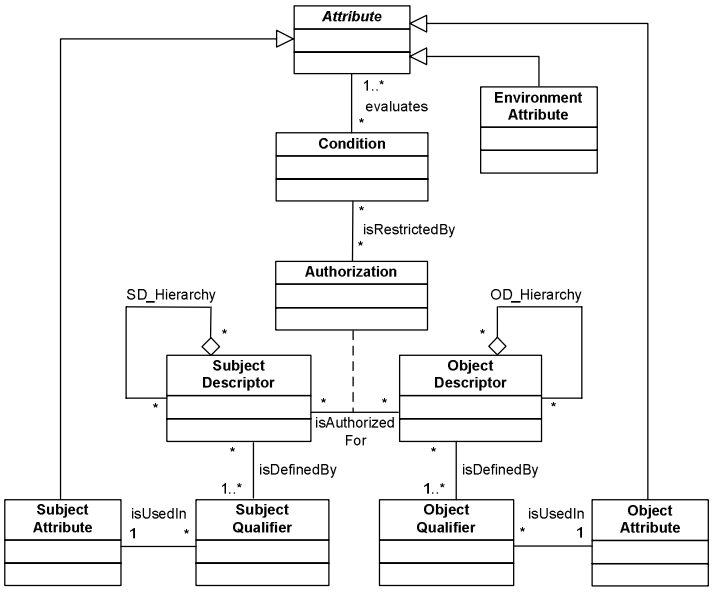


Abbildung 2: Konzeptuelles ABAC-Muster

Neu ist die Möglichkeit, Autorisierungen um zusätzliche Bedingungen zu erweitern. In diesen Bedingungen können zum Einen Subjektattribute mit Objektattributen verglichen werden. Dies ist mit den Deskriptoren alleine nicht möglich, da dort nur Vergleiche auf Basis von konstanten Werten geführt werden. Zum Anderen können Umgebungsattribute wie die Systemzeit oder die Systemlast zusätzlich mit in die Zugriffskontrollentscheidung einbezogen werden, wodurch sich z.B. eine Autorisierung auf die üblichen Geschäftszeiten beschränken lässt.

Autorisierungen werden nun zwischen Subjekt- und Objektdeskriptoren modelliert. Außerdem wird das Konzept des Qualifiers (SubjectQualifier und ObjectQualifier) eingeführt, der für ein Attribut verknüpft mit einer Bedingung steht (z.B. “PLZ beginnt mit 93”). Die Qualifier werden dann zu Deskriptoren zusammengefasst. Ein Deskriptor kann demnach mehreren realen Subjekten bzw. Objekten entsprechen. Subjekt- und Objektdeskriptoren stellen so etwas wie Subjekt- und Objektgruppen dar, nur dass die Zuordnung zu diesen Gruppen nicht explizit, sondern implizit durch Attributwerte erfolgt.

Zur Vereinfachung der grafischen Richtlinien-Spezifikation wurde zusätzlich eine Hierarchisierung der Deskriptoren eingeführt. Dadurch können allgemeine Deskriptoren definiert werden, von denen dann sich nur in einzelnen Qualifiern unterscheidende Spezialisierungen abgeleitet werden können (siehe das Beispiel in Unterabschnitt 3.2).

Die eigentlichen Modellierungselemente für die Zugriffsrichtlinien sind im nun folgenden Profil zusammengefasst; das dargestellte konzeptuelle Muster ist nicht im Sinne eines Metamodells zu verwenden.

UML-Profil. Wie bereits erwähnt, bietet die UML mit den Profilen eine leichtgewichtige

Möglichkeit, spezialisierte Modellierungskonstrukte für bestimmte Anwendungsdomänen bereitzustellen. Da dabei das Metamodell der UML selbst nicht verändert wird, kann die bestehende Syntax und Semantik der UML weiterhin als Grundlage genutzt werden.

Ein UML-Profil ist ein Paket, das als Teil der UML-Infrastruktur definiert ist und dessen Inhalte folglich wie vom ursprünglichen Standard vorgegebene Sprachkonzepte genutzt werden können [HKKR05, S. 335]. Drei Bestandteile können unterschieden werden:

- Stereotypen sind Modellelemente, durch die UML-Metaklassen mit Metaattributen (Tags) und Einschränkungen spezialisiert werden können. Sie sind mit einem Namen versehen und können in einem UML-Diagramm durch textuelle Markierung oder mit einem eigenen Symbol eingesetzt werden.
- Beschränkungen können Vor- oder Nachbedingungen sowie Invarianten für einen bestimmten Stereotyp angeben. Oftmals werden sie in der logikbasierten Object Constraint Language (OCL) angegeben; es ist jedoch auch die Verwendung von natürlicher Sprache oder jeder anderen Programmiersprache möglich.
- Schlüsselwort/Wert-Paare (Tagged Values) sind benannte und typisierte Metaattribute, die einem Stereotyp zugewiesen werden und beliebige Information aufnehmen können.

Wir verwenden Stereotypen hauptsächlich in ihrer einfachsten Form als Klassifizierungsmechanismus für Metaklassen [HKKR05, S. 336]. Die Spezifikation der im Profil enthaltenen Stereotypen findet sich in Tabelle 1. Dort ist neben dem Namen des Stereotyps das jeweils zugehörige Symbol enthalten sowie eine textuelle Beschreibung, die die weitere Definition der Stereotypen vornimmt.

Ergänzend zur Spezifikation der Stereotypen aus Tabelle 1 ist in Tabelle 2 die Verbindung der im ABAC-Muster enthaltenen Klassen zu den definierten Stereotypen gemeinsam mit deren UML-Basisklassen zu sehen. Eine Entsprechung für die Klasse Attribute aus dem Muster ist nicht notwendig, da die Klasse abstrakt ist und in konkreten Richtlinien damit nicht verwendet wird. Die im Muster enthaltene Subjekt- bzw. Objektdeskriptorhierarchisierung (SD_Hierarchy, OD_Hierarchy) wird durch im Richtliniendiagramm durch eine Generalisierung zwischen den jeweiligen Deskriptoren modelliert.

Um ABAC-konforme Klassendiagramme für die Zugriffsrichtlinien zu erreichen, geben wir einige natürlichsprachliche Beschränkungen an; diese sollen die korrekte Benutzung der neuen Elemente sicherstellen:

- Die Stereotypen sind exklusiv auf die Modellelemente anzuwenden, d.h. ein Modellelement darf nur durch einen Stereotyp des Profils stereotypisiert werden.
- Die im Schlüsselwort “Predicate” einer Condition-Assoziationsklasse enthaltene Bedingung darf sich nur auf Attribute beziehen, die mit dieser Klasse assoziiert sind.
- Die einzelnen für die Qualifier angegebenen Einschränkungen müssen sich jeweils auf das assoziierte Attribut beziehen.

Name	Symbol	Beschreibung
SubjectAttribute	SA	Subjektattribute repräsentieren einzelne Eigenschaften von Subjekten des Systems.
ObjectAttribute	OA	Objektattribute repräsentieren einzelne Eigenschaften von Objekten des Systems.
SubjectQualifier	SQ	Subjektqualifier verkörpern einzelne Eigenschaften von Subjekten, die durch einen binären Operator-Operand-Vergleich auf bestimmte Bereiche eingeschränkt wurden. Die Vergleiche werden durch Beschränkungen formuliert, die den Assoziationen zwischen Subjektattributen und Subjektqualifiern zugeordnet werden.
ObjectQualifier	OQ	Objektqualifier verkörpern einzelne Eigenschaften von Objekten, die durch einen binären Operator-Operand-Vergleich auf bestimmte Bereiche eingeschränkt wurden. Die Vergleiche werden durch Beschränkungen formuliert, die den Assoziationen zwischen Objektattributen und Oubjektqualifiern zugeordnet werden.
SubjectDescriptor	SD	Subjektdeskriptoren fassen eine Teilmenge der in Subjektqualifiern enthaltenen eingeschränkten Eigenschaften zusammen und stehen dadurch für Subjekte, die ein bestimmtes Bündel von Eigenschaften aufweisen.
ObjectDescriptor	OD	Objektdeskriptoren fassen eine Teilmenge der in Objektqualifiern enthaltenen eingeschränkten Eigenschaften zusammen und stehen dadurch für Objekte, die ein bestimmtes Bündel von Eigenschaften aufweisen.
Authorization	A	Eine Autorisierung stellt durch eine gerichtete Assoziation dar, dass in einem Subjektdeskriptor beschriebene Subjekte eine bestimmte Operation auf den von einem Objektdeskriptor beschriebenen Objekten ausführen dürfen. Die Assoziation trägt den Namen der Operation.
Environment Attribute	EA	Umgebungsattribute repräsentieren einzelne Systemeigenschaften.
Condition	C	Bedingungen sind zusätzliche Einschränkungen für Autorisierungen. Sie besitzen ein Metaattribut (Tagged Value) “Predicate” vom Typ “string”. Bedingungen können sich auf alle Attributarten beziehen. Sie werden als Assoziationsklasse einer Autorisierung zugeordnet.

Tabelle 1: Stereotypen

ABAC-Modellelement	UML-Metaklasse und Stereotyp	
SubjectAttribute	Class	≪SubjectAttribute≫
ObjectAttribute	Class	≪ObjectAttribute≫
SubjectQualifier	Class	≪SubjectQualifier≫
ObjectQualifier	Class	≪ObjectQualifier≫
SubjectDescriptor	Class	≪SubjectDescriptor≫
ObjectDescriptor	Class	≪ObjectDescriptor≫
Authorization	Association	≪Authorization≫
EnvironmentAttribute	Class	≪EnvironmentAttribute≫
Condition	Class	≪Condition≫

Tabelle 2: Mapping zwischen konzeptuellem ABAC-Modell und UML-Profil

- Die im Muster in Abb. 2 bei den Assoziationen angegebenen Kardinalitäten sind in den Richtliniendiagrammen zu berücksichtigen.

3.2 Ein Beispielszenario

Ein exemplarischer Anwendungsfall soll den Einsatz des Profils verdeutlichen. Eine Wirtschaftsauskunftei mit entsprechenden Befugnissen (z.B. die Schufa) betreibt im Internet einen Dienst in Form eines Web Services, der Informationen über Merkmale und Scorewerte von Personen anbietet. Benutzer können Kreditinstitute sowie natürliche Personen sein. Kreditinstitute sollen im Rahmen von Bonitätsprüfungen Merkmale beliebiger Personen abfragen können. Desweiteren sollen durch die beteiligten Institute als Partner der Auskunftei Merkmale von Personen hinzugefügt bzw. aktualisiert (verändert) werden können. Außerdem sind Personen berechtigt, bei der Auskunftei die sie betreffenden Merkmale abzufragen. Kreditinstitute, die besondere Verträge (Goldstatus) mit der Auskunftei haben, können zudem eine Kundenbewertung in Form eines Scorewerts anfordern. Die angegebenen Beschränkungen sollen zudem nur für Personen und Institute aus Deutschland angewendet werden. Abbildung 3 zeigt den grafischen Entwurf demgemäßer Richtlinien für das vorliegende Szenario.

Wie aus der Abbildung ersichtlich, werden die stereotypisierten Klassen der Übersichtlichkeit halber in einer verkürzten Notation ohne Attribute bzw. Operationen gezeichnet.

Insgesamt sind drei Subjektattribute (Ort, Typ, Status) modelliert, die über mit Einschränkungen versehenen Assoziationen den Anforderungen entsprechende Qualifier definieren. Die Qualifier wiederum werden so mit Subjektdeskriptor-Klassen assoziiert, dass die gewünschten Eigenschaften der Web-Service-Benutzer zusammengefasst werden. Im Einzelnen resultieren ein Deskriptor, der für Personen aus Deutschland steht (Person_DE), sowie zwei Deskriptoren, die für deutsche Kreditinstitute mit bzw. ohne Goldstatus stehen. Zu beachten ist hier die Verwendung der Hierarchisierung der Deskriptoren: Der Deskriptor für Kreditinstitute mit Goldstatus ist vom allgemeinen Kreditinstitut-Deskriptor abgeleitet, so dass nur noch eine Assoziation mit dem Hat_Goldstatus-Qualifier nötig ist.

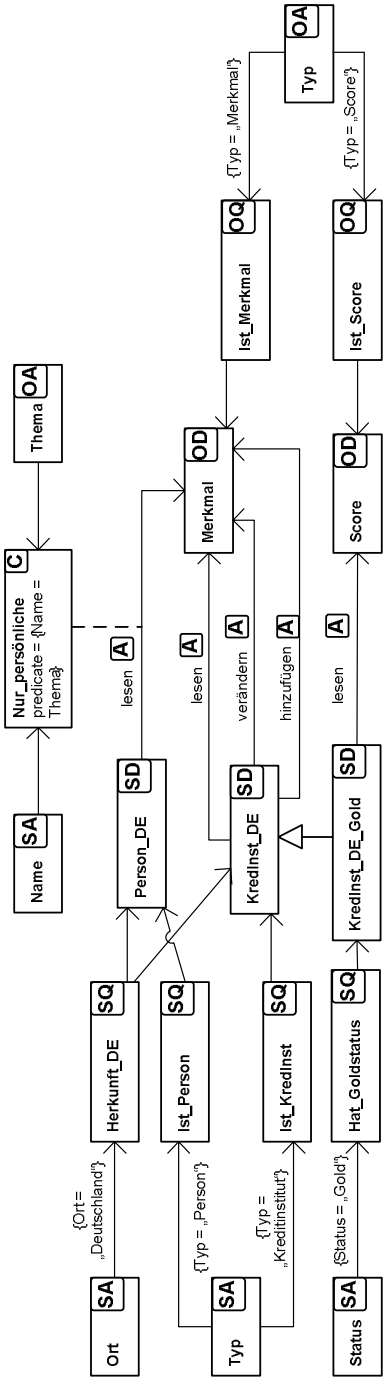


Abbildung 3: Zugriffsrichtlinien für die Auskunftei

Ähnliches zeigt sich auf der Objektseite; hier finden sich zwei Objektdeskriptoren für Personenmerkmale bzw. -scores, die sich nur in ihrem Objekttyp unterscheiden. Die Deskriptoren werden über ein entsprechendes Objektattribut und passende Qualifier gebildet.

Die Autorisierungen selbst sind über gerichtete Assoziationen modelliert, die jeweils mit dem erteilten Zugriffsrecht benannt sind.

Die Umsetzung der Anforderung, daß jede Person die eigenen Merkmale lesen können soll, verdeutlicht die Leistungsfähigkeit des ABAC-Modells. Dies wird hier durch eine Condition-Assoziationsklasse erreicht, die als Schlüsselwort/Wert-Paar ein Prädikat enthält, das das zusätzliche Subjektattribut Name mit dem Objektattribut Thema (bezeichnend den Gegenstand eines Objekts, hier also die Person, die ein Merkmal aufweist) vergleicht und mit ihnen assoziiert ist. Das Metaattribut bei den Condition-Klassen wird entgegen der in [HKKR05, S. 340] beschriebenen Notation aus Gründen der Übersichtlichkeit im obersten Abschnitt des Klassensymbols hinterlegt. Die Ausführungen in Abschnitt 2 wieder aufgreifend, illustriert diese Anforderung die Grenzen des rollenbasierten Zugriffskontrollmodells, das für jede Person eine eigene Rolle bereitstellen müsste.

4 Verwandte Arbeiten

Die grafische Modellierung von Sicherheitsaspekten von Anwendungssystemen allgemein und Zugriffskontrolle im speziellen ist ein aktuelles und vielbehandeltes Thema. Der nachfolgende Überblick beschränkt sich auf Arbeiten, die ebenfalls die UML zur Modellierung verwenden.

Eine Methode zum grafischen Entwurf von sicheren Data Warehouses bietet [VFMP06]; sie bietet ein UML-Profil sowie eine Erweiterung von OCL, um Sicherheitsbeschränkungen wie z.B. Autorisierungsinformationen in Klassendiagrammen von multidimensionalen Data Warehouses repräsentieren zu können. Dieser Ansatz ist für unsere Zwecke ungeeignet, da keine Separation von Daten und Zugriffsbeschränkungen erfolgt, so dass die Sicherheitsinformationen nicht in einem Web-Service-Umfeld verwendet werden können.

Jürjens präsentiert in [Jür02] ein umfassendes UML-Profil namens UMLsec zur Entwicklung sicherer Systeme. Dieses Profil kann Anforderungen u.a. der Vertraulichkeit, des Informationsflusses und der sicheren Kommunikation in verteilten Systemen abbilden. Darüber hinausgehend wird für eine Untermenge der UML eine formale Semantik angegeben, mit der eine Evaluation der Modellbildung durchgeführt werden kann. Jedoch können keine dedizierten Zugriffsrichtlinien modelliert werden.

SecureUML von Basin et al. [BMPP06] ist ein modellgetriebener Ansatz zur Entwicklung von sicheren Systemen. Dabei werden die in der Systemspezifikation angegeben Sicherheitsanforderungen zu Programmcode einer Zugriffskontrollinfrastruktur umgesetzt. Dies geschieht auf der Basis eines erweiterten RBAC-Modells, das zwar zusätzliche Autorisierungseinschränkungen in OCL formulieren kann - ähnlich der Conditions in ABAC -, jedoch nicht die volle Ausdruckskraft sowie Loslösung von Identitäten, wie sie für offene Systeme notwendig ist, erreicht.

Ebenfalls im RBAC-Umfeld bewegt sich [RLFK04], die ein UML-Referenzmuster für die Anwendungsentwicklung mit RBAC angeben sowie beschreiben, wie Beschränkungen zur Aufgabentrennung grafisch modelliert und validiert werden können. Eine explizite Richtlinienspezifikation erfolgt nicht.

In [SAGM05] wird ein Ansatz vorgestellt, der unter Verwendung eines Tools die Elemente des RBAC-Modells nachbildet sowie weitere RBAC-spezifische Beschränkungen in OCL beschreibt. Die so dargestellten Richtlinien können mit dem genannten Tool validiert werden.

Der Vorteil des vorliegenden Ansatzes gegenüber den in diesem Abschnitt vorgestellten Arbeiten liegt zusammengefaßt darin, dass nur mit unserem Profil attributbasierte Richtlinien modelliert werden können, die für offene System geeignet sind.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde eine Erweiterung der UML in Form eines Profils präsentiert, das zur Modellierung von attributbasierten Zugriffsrichtlinien dienen kann. Vorangehend wurde motiviert, warum im Umfeld von Web Services bzw. allgemeiner offener Systeme ein solches Zugriffskontrollmodell notwendig ist, warum eine separate, von den Daten losgelöste Spezifikation der Zugriffsrichtlinien sinnvoll ist und weshalb eine weitverbreitete grafische Notation wie die UML die geeignete Grundlage für ein solche Spezifikation sein kann. Abrundend wurde der Einsatz der entwickelten Notation an einem Beispiel erläutert. Insgesamt erlaubt der dargestellte Ansatz, Zugriffsrichtlinien einheitlich und durch die grafische Darstellung auch dem nicht sehr geübten Administrator zugänglich zu spezifizieren.

Für zukünftige Arbeiten bieten sich mehrere Richtungen an. Ein wichtiger erster Schritt besteht in der Implementierung des Tool-Supports für das Management der Zugriffsrichtlinien. Die vorgestellte Notation soll dafür als Grundlage dienen. Dies schließt eine Funktionalität mit ein, die die spezifizierten Richtlinien ausgehend von XMI, dem XML Metadata Interchange der UML, auf Grundlage eines Transformationsschemas weitestgehend automatisiert in Sprachen wie RuleML oder XACML übersetzt. Diese XML-Daten können danach von einem Zugriffskontrollmodul (das die Funktionen von PDP und PEP übernimmt), wie es z.B. in [BMPP06] vorgestellt wurde, verarbeitet werden. Daneben streben wir die Definition von Funktionalitäten für die Administration von ABAC-Richtlinien an, wie es in ähnlicher Weise in [DMP04] für RBAC beschrieben wurde. Schließlich wollen wir eine Praxiserprobung und Evaluation durchführen. Dies soll im Umfeld des EU-Projekts Access-eGov[3] geschehen, das sich die Entwicklung einer semantischen Web-Service-Infrastruktur mit Peer-to-Peer-Elementen für das eGovernment zum Ziel gesetzt hat. Es ist beabsichtigt, die vorliegenden Ansätze bei der Entwicklung der Sicherheitsinfrastruktur zu verwenden.

[3] `http://www.access-egov.org`

Literatur

[BMPP06] David Basin, Jürgen Doser und Torsten Lodderstedt. Model Driven Security: From UML Models to Access Control Infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15(1):39-91, 2006.

[BMPP06] Sönke Busch, Björn Muschall, Günther Pernul und Torsten Priebe. Authrule: A Generic Rule-Based Authorization Module. In *Proceedings of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Seiten 267-281, 2006.

[DDCdVS05] E. Damiani, S. De Capitani di Vimercati und P. Samarati. New Paradigms for Access Control in Open Environments. In *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2005)*, Seiten 540-545, 2005.

[DMP04] Fredj Dridi, Björn Muschall und Günther Pernul. Administration of an RBAC System. In *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37)*, 2004

[FSG+01] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn und Ramaswamy Chandramouli. Proposed NIST Standard for Role-based Access Control. *ACM Transactions on Information and System Security*, 4(3):224-274, 2001.

[HKKR05] Martin Hitz, Gerti Kappel, Elisabeth Kapsammer und Werner Retschitzegger. *UML@Work*. dpunkt, 3. Auflage, 2005.

[Jür02] Jan Jürjens. UMLsec: Extending UML for Secure Systems Development. In *Proceedings of the 5th International Conference on the Unified Modeling Language (UML 2002)*, Seiten 412-425, 2002.

[KMPP05] Manuel Koch, Luigi V. Mancini und Francesco Parisi-Presicce. Graph-based specification of access control policies. *Journal of Computer and System Sciences*, 71(1):1-33, 2005.

[OAS05] OASIS. eXtensible Access Control Markup Language (XACML) Version 2.0, 2005.

[PDMP05] Torsten Priebe, Wolfgang Dobmeier, Björn Muschall und Günther Pernul. ABAC - Ein Referenzmodell für attributbasierte Zugriffskontrolle. In *Tagungsband der 2. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik (Sicherheit 2005)*, Seiten 285-296, 2005.

[PFMP04] Torsten Priebe, Eduardo B. Fernández, Jens Ingo Mehlau und Günther Pernul. A Pattern System for Access Control. In *Proceedings of the 18th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Seiten 235-249, 2004.

[RLFK04] Indrakshi Ray, Na Li, Robert B. France und Dae-Kyoo Kim. Using UML to Visualize Role-based Access Control Constraints. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, Seiten 115-124, 2004.

[SAGM05] Karsten Sohr, Gail-Joon Ahn, Martin Gogolla und Lars Migge. Specification and Validation of Authorisation Constraints Using UML and OCL. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS 2005)*, Seiten 64-79, 2005.

[VFMP06] Rodolfo Villarroel, Eduardo Fernández-Medina und Mario Piattini. A UML 2.0/OCL Extension for Designing Secure Data Warehouses. *Journal of Research and Practice in Information Technology*, 38(1):31-43, 2006.

[YT05] Eric Yuan und Jin Tong. Attribute Based Access Control (ABAC) for Web Services. In *Proceedings of the 2005 IEEE International Conference on Web Services (ICWS 2005)*, Seiten 561-569, 2005.

[ZRG04] Nan Zhang, Mark Ryan und Dimitar P. Guelev. Synthesising Verified Access Control Systems in XACML. In *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering (FMSE 2004)*, Seiten 56-65, 2004

Approaching a Methodology for Designing Composite Applications Integrating Legacy Applications Using an Architectural Framework

Helge Hofmeister*
ecoware Gesellschaft für Informatik mbH
Bamberg, Germany
Email: hofmeister@ecoware.de

and

Guido Wirtz
Distributed and Mobile Systems Group
Otto-Friedrich Universität Bamberg
Bamberg, Germany
Email: guido.wirtz@wiai.uni-bamberg.de

Abstract: In this paper, an approach toward a problem-oriented design methodology for building composite applications is proposed. This methodology allows for a business process based design that focuses on re-using both existent enterprise services and legacy functionality. The methodology takes a reference architecture into account that describes how composite applications can be structured by means of layers and which analysis and design patterns can be identified and applied on these layers. The presented methodology aligns its phases on the architecture's single layers and on the dependencies that do exist between the patterns of the single layers. These patterns are described as well.

1 Introduction

Service oriented architecture (SOA) is an emerging spirit that promises to allow agile software development by re-using existent functionality. The software applications that re-use functionality are called composite applications.
Inherent aspects of a SOA that should allow for such a re-assembling of functionality are transparent distribution of functional entities, modularization and facilitated (re-)ordering of functions. In this world, distributed functions are called services. Transparent distribution and modularization are of course existent and exploited for quite a time — even if

*Contact

the modularization of services in terms of granularity is different e.g. to object orientation. So the promise of this approach lies in the facilitated composition of services that comes with arising standards as [ACH+05] or [OAS03] that are orchestration mechanisms for web services. Together with the SOA and the first service orchestration languages, people started thinking of applying some sort of business oriented protocol (such as business processes) to basic services and having automated support for the business process right away. We believe that it requires more than technical standards to make this dream come true. Furthermore, requirement analysis as well as the development should be structured by reference architectures and development methodologies.

The quality of a system's design is partially determined by the quality of the basic components' design. Thus, it is important not only to compose services but also to compose well-designed services. This is why there do exist already sophisticated definition processes (such as [Feu05]) or quality metrics (such as [Rei03]) that support the grouping of functionality into services. But despite of the advantages of these approaches, services are, no matter how good they are designed in terms of functional or informational cohesion or the degree of inter-service coupling, by definition distributed across organizational borders, defined by different people and executed by different agents. So it is not realistic to expect well-designed services that are easily to be orchestrated by a business process that is translated into a service orchestration language. Although we consider the SOA approach and all the benefits of its agility as promising, we believe that business requirements that are expressed by business processes should not be restrained by the incompatibility of services. As we derived our initial principles from the analysis of case studies, we consider the heterogenity of application landscapes as a crucial point – even if the landscape is wrapped using services. Hence, our aim is to provide mechanisms that allow for a requirement engineering that is not constrained by the definition of existing services.

After an overview of related work in section 2, the reference architecture and its relevant patterns are outlined in section 3. Afterwards the design phases of the design methodology are presenteded.

2 Related Work

There do exist a lot of software development methodologies, such as the Rational Unified Process [JBR99] or — still notably — the waterfall model [Roy87]. However, these methodologies describe software development methodologies while we focus here only on the aspect of designing the blueprint for the software.
The workflow patterns presented by van der Aalst et al. [vdAtHKB03], the workflow data patterns by Russel at al. [RtHEvdA05], the service interaction patterns by Barrows et al. [BDtH05] as well as the enterprise integration patterns by Hohpe and Woolf [HW04] are the patterns that can identified at the architecture's layers.
Although not working with patterns but in terms service design, the work of Reijers [Rei03] and Feuerlicht [Feu05] provide means for constructing services. The main dif-

ference between the presented work and these service definition methodologies is that the presented methodology focuses on re-using functionality of an existent application landscape rather than adding new services.
A comprehensive introduction to service oriented architectures is given in [PG03].
Multi-layered reference architectures are of course used as well in the area of service oriented design. Notably is the work of Decker [Dec05] who also describes an intermediate layer between business processes and application services that align process and IT. Whilst this work focuses on the semantical gap between processes and services, this is one among several aspects one of the reference architecture used here.
In the area of mapping workflow descriptions to each other Dehnert and van der Aalst developed an approach to map business process descriptions onto workflow descriptions using petri-nets [DvdA04]. This work is complementary to our work as they describe how to derive the fourth level of business processes that are executable by means of workflows.
A methodology for migrating legacy applications is given in [WLB+97]. The butterfly methodology is not focusing on integrating legacy systems but on migrating them while operating the two worlds in parallel. Anyway, some aspects of this approach such as the Chrysaliser for data migration decribe similar concepts as the propsed reference architecture.

3 Architectural Framework for Composite Applications

According to Linthicum, Business Process Integration Oriented Application Integration (BPIOAI) provides another layer on-top of existent system integration concepts such as information oriented application integration (IOAI) or service-oriented application integration (SOAI) (cf. [Lin04]). This means, that system-integration focused technologies such as e.g. JMS messaging (for IOAI) or HTTP-based web services (for SOAI) are controlled by a top-level orchestration layer that implements the business logic. Often, to this business-process implementation, it is referred to as composite application. While the composite application implements the business logic, the used integration technologies solely provide the means for calling application systems that in turn provide the logic that is orchestrated by the composite application. To this part, not implementing any business logic, it is referred to as coupling system.
In this chapter we present a reference architecture for composite applications that leverages the development methodology. The framework is structured by five layers that provide different abstraction levels for composite applications. An overview of the architecture is given in figure 1.

3.1 Layer Zero - Legacy Application Systems

Composite applications reorganize services as they are provided by application systems in order to meet specific requirements. The application system's services are often predefined

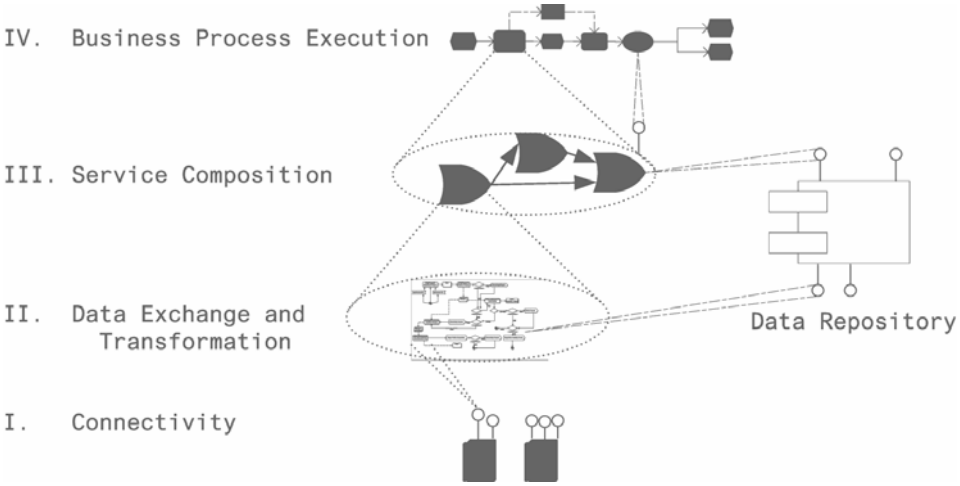


Figure 1: Architectural Layers

by the system's vendor. Alternatively, it might be required to define new services of an application system or to develop new agents that provide new services. As a prerequisite, service oriented architectures need to rely on a common protocol that is shared both by service consumers and providers (cf. eg [Lin04][p. 218]). How application services are technically connected to service consumers, is described at the subsequent layer.

3.2 Layer One - Connectivity

In order to allow the usage of application system functionality in composite applications, this functionality needs to be exposed in a common way. This exposure is described at this layer of connectivity. Here, multiple state transitions of the connected application system are exposed as services. So this layer provides connectivity in between the legacy application system and the composite application by homogenizing the protocol that is used to call functions. From the composite application point of view, this layer provides the application services. Since this layer solely assures connectivity, the actual data format at this layer is still dependent to the connected systems. The exchange and conversion of data is provided at the subsequent layer.

3.3 Layer Two - Data Exchange and Data Transformation

This layer is dedicated to deal with technical complexity by integrating heterogeneous application systems and providing homogeneous interfaces to more high-level functionality. Since we do not mix up business logic with data exchange and transformation functionality here, the probability for re-using functionality from this layer is higher. In turn, subsequent layers do not need to deal with this sort of technical issues. Additionally this

layer unifies the data format of the connected application systems to a canonical data format (cf. [HW04][pp. 355-360] or [Kau90]). After unification of the data, the data is stored into a data repository that is part of our reference architecture as well. This data repository provides the context to the processes that are defined at this and higher layers. Using this context, all services of a composite application can access and exchange data. Additionally this layer provides means for validity checking of data in terms of syntax and semantics as well as error handling procedures that need to be invoked whenever errors occur on this layer[1]. Hence, it provides the functionality that encapsulates the actual communication semantics with the connected systems. This means for instance that acknowledgements for asynchronous calls are transparently handled by this layer. The technical routing is performed here as well.
All the functionality that is provided by this layer is encapsulated in so-called integration services (IS). These services are in turn orchestrated to two different integration processes. One providing data to the upper layers — the integration in-flow (IIF) — and one for publishing data from upper layers to the connected legacy systems. The latter process is called integration out-flow (IOF).
For a more detailed description of the integration services, integration flows and patterns at this layer you might consider [HW06].

3.4 Layer Three - Service Coordination

From a top-down perspective the integration flows provide together with the connectivity services at layer one two standardized services for calling services at or consuming services from underlying application systems. This is irrespectively of communication or computational semantics and provides homogeneous data access as well.
The functionality that is provided by these services is nevertheless determined by the functionality offered by the application systems. It might be necessary to aggregate the application services to more problem-oriented services (enterprise services). At this layer we see the coordination of two to *n* application services with a flow in order to form the enterprise services out of application services.
Conceptually, this coordination layer is recursive. This means that an aggregated service that is composed at this layer might be orchestrated together with services from this or lower layers to form other high-level services at this layer.
Besides for the sake of bringing together requirements and application services, coordination at this layer can also be applied to integrate relevant choreographies into the implementation of a business task.
Ensuring proper state transitions is also matter to this layer's coordination. Rather short-term transactions fulfilling the ACID properties as well as long-running transactions might be required by a composite application.
According to Grefen [GVA01], the transactional coordination consists in the so-called *local transactions* with ACID properties and one the *global transactions* with relaxed trans-

[1] The error handling at this layer basically covers support procedures that need to be initiated whenever errors occur (mostly human errors are the cause for this sort of errors).

actional properties. The latter one uses ACID transactions as black-boxed functionality to form long-term global transactions.
In our framework we meet these ideas by assigning local transactions as being encapsulated at this layer into the compositions of the enterprise services that are exposed by this layer to the business process layer. So the local transaction layer is completely located at our coordination layer and the coordination protocol (such as the 2-phase commit) is fully implemented here.
Meeting the long-term characteristics of global transaction, according to Grefen et al. isolation is relaxed by publishing intermediated results to the global context. Atomicity is relaxed by introducing compensation transactions. Both context publication and compensation transactions are local transactions. Safepoints are local transactions that are marked by this special property of being a safepoint. Thus, the fundamental support for global transactions is formed by local transactions.
[GVA01] proposes as well a way of specifying transactional properties (such as the safe point properties for local transactions) and an execution model that supports global transactions based on these specifications.

The service choreography that is incorporated into this layer, describes how the external services and the service that is formed by the layer interact. In order to analyze the interactions we used the service interaction patterns by Barrows et al. [BDtH05].

Decker introduces a so-called process support layer that deals with various aspects of incompatibility between business process tasks and application services [Dec05]. To a certain extent some of these ideas are supportive in terms of designing the third layer. Anyway, the process support layer cannot be mapped directly to our layer of service coordination but most of the referenced patterns are relevant here as well. These are the patterns for granularity problems as well as the patterns for interdependency problems.

The third purpose of the service coordination layer is to leverage transactions between multiple application services. As transactional interaction is a requirement, these requirements have to be included into the design at this layer. In order to capture this design knowledge as well by the means of patterns we introduce two design patterns as they are informally included in Grefen's work [GVA01]: Local Transaction Composition (LTC) and Global Transaction Composition (GTC).

In order to support these two transactional patterns, we require three properties for services or transactional compositions as mandatory. These are: Safepoint, Idempotent and Compensation. Note that the *Safepoint* property is unary whilst *Idempotent* and *Compensation* are binary properties between transaction compositions. These properties need to be assigned to the single services of a GTC in order to allow the calculation of GTCs' compensations.
While an LTC is a black box and commits changes to the context of a workflow after a successful completion, GTCs are more interactive in terms of context updates. In order to describe the interactions with the context, the workflow data patterns from [RtHEvdA05] are referenced as design patterns at this layer as well.
In this context the data visibility patterns are used to determine how single services of one transaction compositions share their data. The data exchange between transaction compositions is described by data interaction patterns. The data transfer patterns can describe the

mechanism of how contexts that are only accessible by a transaction composition can be made available to other services.

3.5 Layer Four - Business Processes

The 5th layer is the layer where business processes are to be executed using workflow models. The actual functions of the according business processes are realized as services that are exposed by the lower layers. At this layer it is of importance important how process branching based on certain indicators (like states) might be performed. Thus, the process environment needs to have access to the actual context of the process. The data transfer between application services and the actual context is realized by the Data Exchange and Data Transformation Layer. The visibility of data is controlled by the Service Coordination layer and transactional properties are incorporated into the process environment this way. Access to the process context is realized by the services that are provided by the coordination layer.

In order to analyze, design and automate business processes with regards to composite applications, the control as well as the data perspective are important at this layer (cf. [vdAtHKB03]). This is because the sequence of underlying tasks that are represented by services is determined at this layer.
For the perspective of control flows there exist a catalogue of so-called workflow patterns. The work of van der Aalst et al. consists in a set of patterns that are distinguished into basic control flow patterns, advanced branching and synchronization patterns, patterns involving multiple instances, state-based patterns and cancellation patterns.
For the data perspective there exists a broad set of patterns as well. [RtHEvdA05] distinguishes the data patterns into patterns for data visibility, data interaction, data transfer mechanisms and data based routing. Solely the category of data based routing pattern is relevant for this layer as these patterns describe how data needs to be accessed in order to guarantee the execution of workflows. As other aspects are not covered by this layer, are the related patterns also relevant for other layers.
All these patterns are artifacts that can be identified within a workflow description[2]. Thus, we consider the mentioned sets of patterns as analysis patterns that can be used in later phases on lower layers in order to support the design ibidem.

4 A Methodology to Design Composite Applications

The methodology is closely related with the layers and patterns that have been introduced in the previous section. Thus, describing the methodology basically consists of discussing the phases that are required to step through that reference architecture.

[2]Of course, they are also useful for designing engine-specific workflow descriptions out of business process descriptions.

4.1 Business Process Modelling and Requirements Engineering

First of all, our approach is very business process centered. Thus, it is absolutely required to model the relevant business process using a methodology of choice. Important is that the modelled business process describes not only the control flow perspective. Furthermore, it is required to also describe the data, organizational and operational perspective [3]. Having this process description as an input, the phases that are described below can be applied.

4.1.1 Enterprise Service Matching

Having described the functional requirements of a to-be built composite application by expressing the control and data flow perspective of an business process, the single process steps — usually referred to as functions — need to be matched with services. In order to support the execution of these functions, the operational perspective needs to be checked in terms of finding application systems that might already expose suitable services. *Suitable* in that case means that the service offers operations that *match* the functional requirement. This requirement is usually expressed in a non-formal (usually verbal) way in terms of required functionality, and in terms of the process's data perspective. The data perspective here describes which input and output format are to be sent/received by the identified service.[4]
Whenever there is no suitable service in the organization's service repository, but the operational perspective of the business process refers to one or more application services, the functionality of these application systems need to be analyzed in terms of finding functional areas that might be appropriate for realizing the required functionality. In large organizations, usually a fraction of the required functionality is already existent in legacy systems.
Anyway, sometimes there might not exist any application system at all, and the business process function needs to be implemented from scratch.

The artifact of this phase is either the enterprise service identified in the repository, the functional areas of relevant application systems that might be used to implement the enterprise service or the information that a new service is required to be developed.

4.1.2 Data Interaction Analysis

The next step is to analyze the data interaction as it is roughly described in the data perspective and in the control flow perspective of the business process. This can be done by identifying data interaction patterns' (cf. [RtHEvdA05]) data-based routing patterns within the process. This information is required in order to identify which data needs to

[3] Note that the operational perspective is — especially for new systems — often incomplete at the time of requirements engineering.

[4] Even if a lot of work has provided means for matching services, even dynamically at runtime, we do not consider these possibilities yet, since large organizations usually lack a formal description of their legacy systems' functionality. The same is currently true for more recently built service repositories.

be accessed by the process execution layer in order to determine the appropriate process branch.

The artifacts of this phase are the identified patterns as well as the relevant part of the data-model the pattern operates on as well as specific attributed of the patterns such as target values or branching conditions.

4.1.3 Transactional Property Identification

Identifying transactional requirements in the business process is a rather difficult task. Therefore, several iterations are required to gather a complete picture. As a first step, it is required to express the already known transactional spheres in the business process. These might either be a set of functions[5] that need to be grouped into one ACID transaction, or more probable, sets of functions that need to be grouped into *global transactions* with relaxed ACID properties.

The artifacts of this phase are the subgraphs of the business process and the information they should be handled as *local* or *global* transactions.

4.1.4 Definition of Non-Functional Requirements

In the next phase, non-functional requirements such as required response times have to be collected for the single functions of the business process. An artifact of this phase could be e.g. a catalogue as it is described in [BSDL06].

4.1.5 Service Composition

This phase marks the transition from the business process layer to the service composition layer. Here it is identified whether certain business protocols are required to be respected while supporting single business process functions. The part of the protocol that is required to be implemented by the actual organization has to be expressed by service interaction patterns (cf. [BDtH05]). Eventually, other services than the already identified are involved in such a protocol. This phase finalizes the description of the service coordination layer's interactions.

The artifacts of this phase are the references to the involved services as well as the description of their interactions by the means of the service interaction patterns.

4.1.6 Application Service Determination

At this stage the design decision is made how the (eventually) identified lack of enterprise service can be overcome. For that sake the application systems that have been identified

[5] This set might also solely contain a single function.

during the *enterprise service matching*-phase needs to be refined in terms what functionality can be combined in order to form the required enterprise service. For that sake, potential functions of legacy systems, service interaction patterns and the patterns of [Dec05], need to be used in order to describe the interaction between the legacy systems. In order to not modify standard software, also functions from the same legacy system might be required to (re-)composed by the interaction described in this phase.

The artifact of this phase are the detailed description of the legacy functions as well as the description of their composition by means of the named patterns.

4.1.7 Transactional Composition Analysis

Taking the information from the first transactional analysis phase into account, this phase uses this information together with the complete description of the service coordination layer in order to complete the picture of transactional properties.
Again, this involves the identification of workflow sub-graphs in terms of them being *local* or *global transactions*. Additionally, single steps or complete *transactions* need to be categorized. This involves assigning the properties *Safepoint*, *Idempotent* and *Compensation*. The *Safepoint* property is unary and therefore only to be assigned to a single transaction, *whilst Idempotent* and *Compensation* are binary properties between *transactions*. These properties can later on support the runtime to calculate compensations which is necessary to allow for long-running transactions with relaxed ACID properties.

The artifacts of this phase are both the identification of *transactions* in terms of their composition sub-graph as well as their properties.

4.1.8 Top-Down Data Repository Design

With the yet produced artifacts, it is possible to determine which data in the data repository has to be accessed when and by which component. Additionally, it is determined, how these accesses need to be protected in terms of transactional properties. As a result, in this phase it is possible to describe the interaction with the data repository that establishes the context for the composite applications from a top-down perspective.

The artifacts of this phase is the description of operations that need to be performed against the data repository. This includes the description of the interaction as well as the possibly necessary design of new interfaces and wrappers (especially facades for the transactional access to data entities might be required.

4.1.9 Data Exchange and Data Transformation Definition

Having described the service coordination layer in total, the produced artifacts can be used in order to design the layer of data exchange and data interaction. For that sake the service interaction patterns can be applied in order to constrain the patterns that can be used in the design of this layer. Tables 1 and 2 describe how the service interaction patterns constrain

Interaction Pattern	IIF	Sync	Async	Event	Create MI	Select Cons	Polling Cons	Validity	Ack Valid	Refuse Inval	Receiver	Ack after Rec	Msg Seq.	Splitter	Aggreg	Resequencer	Store Context	Transformator	Ack aft Forw
1. Send	~																		
blocking																			
non-blocking																			
Party unknown																			
Party known																			
2. Receive	X			X							X						X		
Reliable Receiver																			X
Discarding Msg						X													
3. Send/Receive	X			X		X					X								
PartyKnown																			
PartyUnknown																	X		
Blocking	X																		
1 Continuation																			
2 Continuations					X														
4. Racing Incoming Msg	X			X							X								
different msg types					X														
different processes					X														
discard 2nd message						X													
ranking						X													
non-deterministic						X													
5. One-to-many send	~																		
nr of parties known																			
nr of parties unknown																			
reliable delivery																			
6. One-from-many receive	X			X		X					X				X				
7. One-to-many send/rec	X			X		X					X								
nr of parties known																			
nr of parties unknown																			
reliable delivery																			X
8. Multi Responses	X			X							X								
unique message type																			
different message types					X	X													
stop notification																			
9. Contingent Requests	X				X	X w TimOut					X								
10. Atmoic Multicast ntf	~																		
11. Request with Referral	~																		
refer to s ingle party																			
refer to multiple parties																			
a priori known parties																			
unkwon redirect parties																			
12. Relayed Request	~																		
13. Dynamic Routing	~																		

Table 1: Pattern Mapping Concerning the IIF

this design. Crosses (X) indicate mapping between patterns, swung dashes (˜) indicate that the according pattern cannot be used.

Applying these dependencies leads to a pre-selection respectively to discarding of several patterns of the data exchange and transformation layer.
In order to complete the design, the integration flows that have been identified as being relevant have to be checked in terms of which of its patterns is required. Additionally, all parameters coming along with these patterns have to be described as well. The non-functional requirements are used for some decisions at this point as they further constrain the decisions.

The artifact of this phase is the complete description of the required integration in-flows as well as the description of the integration outflows. Note, that this also defines in detail which legacy systems are integrated into the composite application by which means.

4.1.10 Final Design of the Data Repository with Bottom-Up Aspects

Having specified the second layer — especially the data that flows into/out of the *Store/Fetch Canonical Data* integration services (cf. [HW06]), allows for finalizing the data repositories design from a bottom-up point of view.

As for the top-down iteration, the artifacts of this phase is the description of operations that need to be performed against the data repository. This includes the description of the interaction as well as the possibly necessary design of new interfaces and wrappers (especially facades for the transactional access to data entities might be required.

Interaction Pattern	IOF	Sync	Async	Fetch Context	Pre-Transform	Router	Content Based	Dynamic	Rec. List	Transformator	Updater	Resequencer	Sequence	Aggregator	Cmd Msg	Doc Msg	Event Msg	Ask If Succ.	Ref. On Fail	Error Handling
1. Send	X										X									
blocking		X																		
non-blocking			X																	
Party unknown								X												
Party known																				
2. Receive	~																			
Reliable Receiver																				
Discarding Msg																				
3. Send/Receive	X										X									
PartyKnown																				
PartyUnknown				X				X												
Blocking		X																		
1 Continuation																				
2 Continuations																				
4. Racing Incoming Msg	~																			
different msg types																				
different processes																				
discard 2nd message																				
ranking																				
non-deterministic																				
5. One-to-many send	X										X		X							
nr of parties known																				
nr of parties unknown								X												
reliable delivery																				X
6. One-from-many receive	~																			
7. One-to-many send/rec	X					X					X									
nr of parties known									X											
nr of parties unknown								X												
reliable delivery																		X	X	
8. Multi Responses	X										X									
unique message type																				
different message types																				
stop notification																				
9. Contingent Requests	X										X									
10. Atmoic Multicast nif	X	X									X							X	X	X
11. Request with Referral	X										X						~			
refer to single party																				
refer to multiple parties						X					X									
a priori known parties									X											
unkwon redirect parties								X												
12. Relayed Request	X										X				X	X				
13. Dynamic Routing	X																			

Table 2: Pattern Mapping Concerning the I0F

4.1.11 Connectivity Check

In order to connect the identified legacy systems to the composite applications, this phase checks whether the connectivity (layer one of our reference architecture) is sufficient. This is quite straight-forward since only the idioms for the *Event*, the *Receiver* as well as for the *Updater* service need to be supported.

The artifacts of this phase is the information whether the supplied connectors are sufficient in terms of these services and idioms. If the support for the identified idioms by means of connectors is not sufficient, a dedicated project might be launched that addresses the identified issues.

4.1.12 Service Design

As this methodology focuses on re-using legacy application systems, we do not detail the phase(s) that are required whenever functionality can not be built by (re-)assembling existent functionality. Due to the immense complexity of this topic, only as an place-holder this methodology incorporates a design-phase that has to design missing services.

The artifact of this phase(s) are the design for the implementation of new enterprise or application services.

5 Summary and Outlook

We have presented an approach how a reference architecture can be used to formulate a design methodology that builds on-top of legacy applications. As we strongly incorporate both analysis and design patterns, the outlined methodology allows for a business problem oriented design. Additionally, the reference architecture separates different architectural

aspects into different layers.

Since re-use is crucial here, we plan to focus in future work on improving the process of determining appropriate services/legacy functionality.
Additionally, we need to specify a framework for our reference architecture that provides an execution environment for the designed composite applications. This will include the specification of interfaces between the single architectural layers as well as formalizing the access to the data repository. The latter will provide similar mechanisms as the Service Data Objects (SDO) specification [BBB+05] proposes with their Data Access Service. Since the framework is not existent yet we can only map some aspects of the refernce architecture to existing solutions. Anyway, in order to gain experience we are currently applying the methodology for itself in a industry case study. After completion of the design we are going to realize the design using the SAP Netweaver [SAP] product stack.

References

[ACH+05] Tony Andrews, Francisco Curbera, Dholakia Hitesh, Yaron Goland, Johannes Klein, and Frank Leymann. Web Service Business Process Execution Language for Web Services Version 2.0 - Draft. Technical Report 2.0, OASIS, December 21 2005.

[BBB+05] John Beatty, Henning Blohm, Christophe Boutard, Stephen Brodsky, Michael Carey, and Jean-Jacques Dubray. Service Data Objects For Java Specification. Technical report, BEA and SAP and IBM, 2005.

[BDtH05] Alistair P. Barros, Marlon Dumas, and Arthur H. M. ter Hofstede. Service Interaction Patterns. In Wil M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera, editors, *Business Process Management*, volume 3649, pages 302–318, 2005.

[BSDL06] Taiseera Hazeem Al Balushi, Pdro R. Falcone Sampaio, Divyesh Dabhi, and Pericles Loucopoulos. Performing Requirements Elicitation Activities Supported by Quality Ontologies. In *The 2006 International Conference on Software Engineering and Knowledge Engineering (SEKE'06)*, July 5-7 2006.

[Dec05] Gero Decker. Bridging the Gap between Business Processes and existing IT Functionality. In *Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05)*, 2005.

[DvdA04] Juliane Dehnert and 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.

[Feu05] George Feuerlicht. Application of Data Engineering Techniques to Design of Message Structures for Web Services. In *Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05)*, 2005.

[GVA01] Paul Grefen, Jochem Vonk, and Peter Apers. Global transaction support for workflow management systems: from formal specification to practical implementation. *The VLDB Journal*, 10(4):316–333, 2001.

[HW04] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns*. The Addison Wesley Signature Series. Pearson Education Inc., 2004.

[HW06] Helge Hofmeister and Guido Wirtz. A Pattern Taxonomy for Business Process Integration Oriented Application Integration. In *The 2006 International Conference on Software Engineering and Knowledge Engineering (SEKE'06)*, July 5-7 2006.

[JBR99] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[Kau90] G. Kaufman. Pragmatic ECAD Data Integration. Technical Report 1, New York, NY, USA, 1990.

[Lin04] David S. Linthicum. *Next Generation Application Integration*. Addison-Wesley, Boston, MA USA, 2004.

[OAS03] OASIS Open. Web Services Composite Application Framework (WS-CAF), 2003.

[PG03] M. P. Papazoglou and D. Georgakopoulos. Introduction to Special Issue on SOC. *Commun. ACM*, 46(10):24–28, 2003.

[Rei03] Hajo Reijers. A Cohesion Metric for the Definition of Activities in a Workflow Process. In *CaiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Desing (EMMSAD, 03). Velden, Austria.*, 2003.

[Roy87] W. W. Royce. Managing the Development of Large Software Systems: Concepts and Techniques. In *ICSE*, pages 328–339, 1987.

[RtHEvdA05] Nick Russell, Arthur H. M. ter Hofstede, David Edmond, and Wil M. P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. pages 353–368, 2005.

[SAP] SAP. *SAP Netweaver*.

[vdAtHKB03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[WLB+97] Bing Wu, Deirdre Lawless, Jesus Bisbal, Ray Richardson, Jane Grimson, Vincent Wade, and Donie O'Sullivan. The Butterfly Methodology : A Gateway-free Approach for Migrating Legacy Information Systems. In *ICECCS*, pages 200–205. IEEE Computer Society, 1997.

Eine Integrationsplattform zur Verknüpfung von Geschäftsprozessen und IT-Services

Simon Thalbauer, Josef Küng, Peter Regner, Thomas Wiesinger

Institut für Anwendungsorientierte Wissensverarbeitung (FAW)
Johannes Kepler Universität Linz
Altenbergerstraße 69
4040 Linz
{sthalbauer, jkueng, pregner, twiesinger}@faw.at

Abstract: Informationssysteme haben einen großen Einfluss auf die Geschäftsprozesse von Unternehmen. In den meisten Fällen werden durch die Einführung neuer Informationssysteme Geschäftsprozesse grundlegend geändert oder auch gänzlich neue geschaffen. So versprechen sich Unternehmen mehr Flexibilität und sinkende IT-Kosten nach einer serviceorientierten Ausrichtung ihrer Anwendungen, wobei eine große Herausforderung darin besteht, geeignete Systemarchitekturen und lauffähige Anwendungen zu entwickeln, die an den Geschäftsprozessen ausgerichtet sind und sehr flexibel aufgrund von Änderungen in den Geschäftsprozessen angepasst werden können. Nachdem Geschäftsprozessmodelle im Normalfall nicht von Softwareentwicklern, sondern von Fachleuten aus den unterschiedlichsten Unternehmensbereichen erstellt werden, stellt die Transformation von Geschäftsprozessmodellen in Softwareapplikationen bzw. IT-Services eine große Herausforderung dar. Ziel dieser Arbeit am FAW ist die Entwicklung eines praxistauglichen Konzepts zur Optimierung des Zusammenspiels zwischen Geschäftsprozessmanagement und serviceorientierter Softwareentwicklung. Das Konzept wird durch eine innovative Integrationsplattform unterstützt, die die Werkzeuglücke zwischen Geschäftsprozessmodellierungs- und Softwareentwicklungswerkzeugen schließt und eine Best of Breed Lösung ermöglicht.

1 Einleitung

Der Versuch, Geschäftsprozesse in Softwareprodukte weitgehend automatisiert zu transformieren, stellt eine große Herausforderung dar. Das folgende Kapitel beschreibt grundlegend den Stand der Technik und die damit verbundenen Herausforderungen.

1.1 Stand der Technik

Während Methoden und Werkzeuge für das Management von Geschäftsprozessen einerseits und für die Entwicklung von Softwaresystemen andererseits für ihren jeweiligen Anwendungsbereich ausgereift sind, sind aus Sicht der Autoren beide Welten nur unzureichend integriert. Die existierenden Ansätze, die beide Welten zu vereinen versuchen, weisen Schwächen in jeweils einem der beiden Bereiche auf. Dabei lassen sich zwei Typen von Werkzeugen unterscheiden:

Typ 1: Werkzeuge, die aus dem Bereich der Softwareentwicklung kommen, bieten Unterstützung im Bereich der Geschäftsprozessmodellierung – die dafür zur Verfügung gestellten Modellierungskonstrukte decken die Komplexität der Geschäfts-prozessmodellierung nur unzureichend ab. Softwareentwicklungswerkzeuge, wie zum Beispiel der Rational Architect[1] von IBM, Eclipse[2] erweitert um diverse UML Pug-Ins oder der JBuilder[3] von Borland bieten zwar die Möglichkeit, UML Diagramme zu zeichnen und daraus entsprechend Klassen zu generieren. Teilweise ist es auch möglich rudimentär Geschäftsprozessmodelle in UML zu modellieren. Jedoch stellen diese Tools keinen vollwertigen Ersatz für ein Geschäftsprozessmodellierungstool dar. Darüber hinaus werden keine Funktionen zur Unterstützung des Managements der Geschäftsprozesse, also für die Analyse, Bewertung und Simulation von Geschäftsprozessen angeboten.

Typ 2: Werkzeuge, die aus dem Bereich des Geschäftsprozessmanagements kommen, bieten Unterstützung im Bereich der Softwareentwicklung – diese Unterstützung beschränkt sich auf die grafische Modellierung von Softwaresystemen, in den meisten Fällen auf Basis von UML. Als Beispiel seien hier Adonis[4] der Firma BOC und das Geschäftsprozessmodellierungswerkzeug ARIS[5] der IDS Scheer AG erwähnt, die beide sehr gute Softwarelösungen für den Bereich der Geschäftsprozessmodellierung darstellen. Beide Produkte bieten allerdings keine Möglichkeit die erstellten Geschäftsprozessmodelle in Programmcode zu übertragen.

Thomas Allweyer zählt in seinem Artikel [Al05] verschiedene Methoden auf, wie die Inhalte von Prozessmodellen in Informationssysteme übertragen werden können (Standardsoftware, Workflow, BPM-Tools, Business-Rules-Engine, Model-Driven Architecture, Klassische Software-Entwicklung).

Konzeptionell stellt der in dieser Arbeit gewählte Ansatz eine Erweiterung der modellgetriebenen Softwareentwicklung (Model-Driven Architecture[6]) [MM03] dar, die in der nachstehenden Grafik skizziert wird. Während dieses Konzept die automatische Transformation von Modellen der Softwareentwicklung vorsieht (Transformation von plattformunabhängigen Modellen (PIM) in plattformspezifische Modelle (PSM) und

---

[1] http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html
[2] http://www.eclipse.org/
[3] http://www.borland.com/de/products/jbuilder/
[4] http://www.boc-eu.com/bochp.jsp?file=WP_582571cc1ed802de.46e381.f59775478f.-7f17
[5] http://www.ids-scheer.de/germany/products/53956
[6] http://www.omg.org/mda/

schließlich die Generierung von Code) ist eine Überführung der computerunabhängigen Modelle (CIM), die den Geschäftsprozessmodellen entsprechen, grundsätzlich nur manuell möglich, bzw. nur dann automatisierbar, wenn der fachliche Modellierungsprozess weitgehend formalisiert wird. Die „Überformalisierung“ des fachlichen Modellierungsprozesses, der ja primär von den Fachexperten durchgeführt werden soll, soll nicht verfolgt werden.

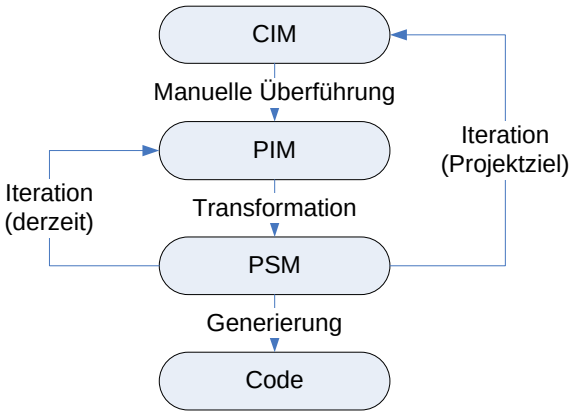


Abbildung 1: Model-Driven-Architecture [in Anlehnung an MM03]

Es wird eine Best Of Breed Lösung angestrebt, bei der Werkzeuge, die in ihrem jeweiligen Anwendungsbereich ausgereift sind, zum Einsatz kommen. Der Austausch der Informationen und somit die Integration beider Welten erfolgt über eine Integrationsplattform (siehe Kapitel 3.1).

1.2 Grenzen einer automatischen Transformation

Fachliche Anforderungen in Form von Geschäftsprozessen können nicht gänzlich automatisch in Softwaresysteme umgewandelt werden können. So findet sich etwa in [SJW05], Aussagen wie „…die fachliche Sicht des Anwenders auf die Abläufe unterscheidet sich oft gravierend von der späteren technischen Implementierung …“ oder „…der kreative Akt des Anwendungsdesigns ist für komplex strukturierte Anwendungen nicht automatisierbar …“.

Einige Informationstypen können nahezu vollständig automatisiert Richtung Softwareentwicklung übertragen werden (z. B. Ressourcen, Organisationseinheiten, Dokumente). Diverse andere Informationen, wie zum Beispiel Geschäftsprozesse lassen sich in den meisten Fällen nur manuell transformieren.

Eines unserer zentralen Ziele ist die bestmögliche Unterstützung des „kreativen Aktes des Anwendungsdesigns“ aus bestehenden Geschäftsprozessen. Grundlage dafür ist die Klassifizierung der Informationen in solche, die automatisiert und solche, die nur

manuell transformiert werden können. Die Integrationsplattform unterstützt die beteiligten Personen bei der manuellen Transformation der Informationen.

1.3 Ineffizienter Ablauf

Ein in der Praxis weit verbreiteter Ansatz der prozessorientierten Anwendungsentwicklung basiert auf der manuellen Übernahme der Geschäftsprozessinformationen in Softwaremodelle und in weiterer Folge in Quellcode. Kommunikationsmittel sind dabei das Lasten- bzw. Pflichtenheft.

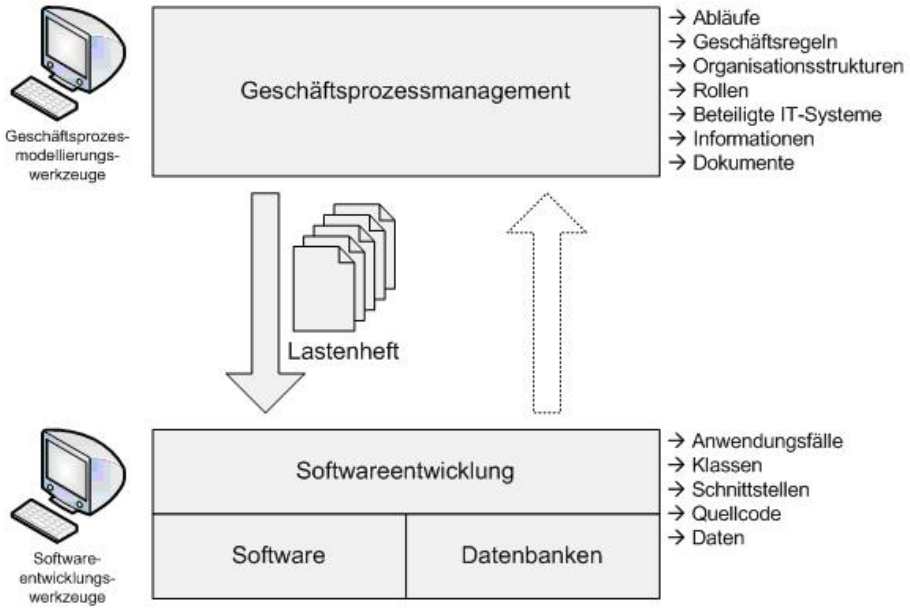


Abbildung 2: Traditioneller Ansatz der prozessorientierten Softwareentwicklung

Zur Kommunikation der Ergebnisse der Softwareentwicklung müssen die geänderten Geschäftsprozesse wiederum manuell erstellt werden. Ein mehrmaliger Abgleich der Informationen aus der Welt der Geschäftsprozesse mit jenen der Softwareentwicklung ist aus Zeit- und Kostengründen meist nicht möglich.

2 Herausforderungen

Um den traditionellen Ansatz der prozessorientierten Softwareentwicklung zu verbessern, arbeitet das FAW mit einem industriellen Partner zusammen, mit dem Ziel, das Konzept zur Effizienzsteigerung des Transformationsprozesses durch den Einsatz geeigneter Werkzeuge kombiniert mit der zu entwickelnden Integrationsplattform zu

optimieren. Im Unternehmen des Projektpartners verwendet zur Modellierung der Geschäftsprozesse das Tool Adonis und zur Implementierung der Softwaresysteme das Programm Rational Architect (siehe Kapitel 1.1. Diese Tools bieten die softwaretechnische Ausgangslage des Projekts und die zu entwickelnde Integrationsplattform fungiert als Bindeglied der angesprochenen Systeme. Im Zuge des Projekts versuchen die beteiligten, die Schnittstellen (Schnittstelle Adonis – Integrationsplattform, Schnittstelle Integrationsplattform – Rational Architect) so allgemein wie nur möglich zu halten, um die Integrationsplattform in weiterer Folge auch in anderen Systemumgebungen einsetzen zu können.

In diesem Kapitel werden die Herausforderungen, welchen sich das Konzept die Methode und die Integrationsplattform zu stellen haben, aufgelistet. Sie wurden in einer ersten Analysephase gemeinsam mit dem Projektpartner erarbeitet.

Änderbarkeit von Geschäftsprozessen:

In [Ku05] vertreten die Autoren die Ansicht, dass Geschäftsprozesse nahezu ausschließlich in Softwareapplikationen fest codiert sind, was als Resultat eine hohe Komplexität der Applikationsstruktur nach sich zieht. Nachdem Geschäftsprozesse und die davon betroffenen Modelle Veränderungen unterliegen können, stellen das Implementieren der Geschäftsprozessmodelle und das ständige Anpassen der Software hohe Anforderungen an die Softwareentwicklung, auch wenn die Geschäftsprozess-modelle äußerst detailliert ausgefertigt wurden.

Modellierungsstil:

Hinderlich für eine möglichst allgemein gültige Lösung, Geschäftsprozessmodelle weitgehend automatisiert in Software zu transformieren, sind die unterschiedlichen Modellierungsstile der einzelnen Geschäftsprozessmodellierer. Vor allem aus diesem Grund stellt es eine Notwendigkeit dar, dass sich die Modellierer an etwaige Modellierungskonventionen halten um einen möglichst einheitlichen Modellierungsstil einzuhalten, wobei es für den Modellierungsprozess sehr wichtig ist, dass die Modellierer durch diverse Konventionen und Regelwerke nicht zu sehr in ihrer Tätigkeit eingeschränkt werden.

Detailliertheitsgrad:

Geschäftsprozessmodelle weisen in der Regel immer unterschiedliche Detail-liertheitsgrade auf. Das Spektrum reicht von sehr allgemeinen, nur die oberste Ebene beschreibenden Geschäftsprozessmodellen bis hin zu äußert detailliert ausgeführten Modellen, die zahlreiche Ebenen der Geschäftsprozesse abbilden und zusätzliche Attribute bzw. Input- und Outputparameter definieren. Je nach Detailliertheitsgrad ändern sich die Anforderungen an die zu entwickelnde Implementierungsplattform, wobei man davon ausgehen kann, dass der Transformationsaufwand zunimmt, je weniger detailliert ein Geschäftsprozess modelliert wurde.

Bei der Erstellung der Modelle wird in den meisten Fällen Top Down vorgegangen. Ausgehend von einer sehr abstrakten Sichtweise des Problembereiches wird diese

schrittweise detailliert. Die Detaillierung im Rahmen der Prozesshierarchie erfolgt in Ebenen. In jeder Ebene wird versucht denselben Abstraktionsgrad für die Teilbereiche zu erzielen. Wird ein Geschäftsprozess detailliert, so steigt man in der Hierarchie eine Ebene tiefer. Die Detaillierung des Problembereichs erfolgt auf Basis von Risikoüberlegungen, in Anlehnung an das Motto, wenn es ein Risiko darstellt, etwas nicht präzise zu spezifizieren, dann ist es zu spezifizieren bzw. wenn es ein Risiko darstellt etwas zu spezifizieren, dann ist es nicht zu spezifizieren. Abbildung 3 veranschaulicht grafisch den soeben ausgeführten Absatz.

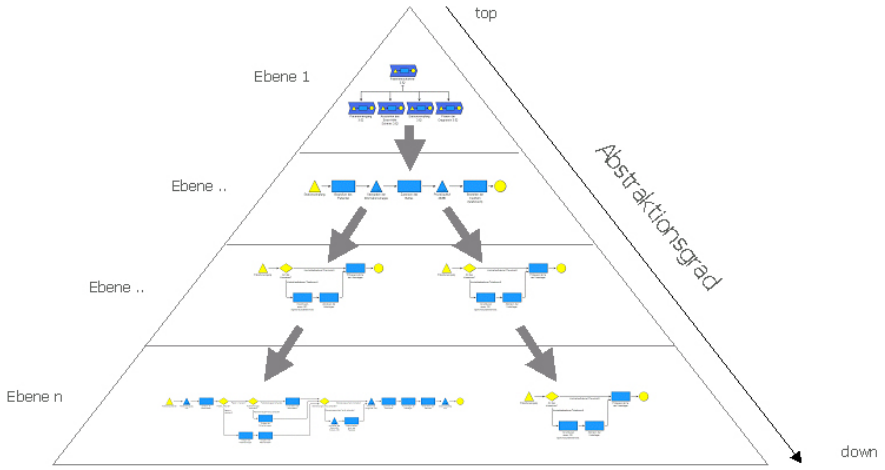


Abbildung 3: Abstraktionsgrad von Geschäftsprozessmodellen („Prozesspyramide“) [RWW03]

Benutzerakzeptanz:

Ein besonderes Risiko stellt die Akzeptanz der Benutzer auf Seiten der Geschäftsprozessmodellierung dar. Die Modellierung der Geschäftsprozesse wird primär von Mitarbeitern aus Fachabteilungen durchgeführt, die in den seltensten Fällen Erfahrungen aus dem Bereich der Softwareentwicklung aufweisen. Die Erfahrung hat gezeigt, dass umfangreiche Einschränkungen der Freiheitsgrade bei der Modellierung die Qualität der Ergebnisse negativ beeinflusst. Ziel ist es daher, ein akzeptables Maß an Vorgaben zu entwickeln, welches die fachliche Modellierung nicht zu sehr einschränkt, trotzdem aber für die Softwareentwicklung verwertbare Informationen geliefert werden können. Diesem Risiko wird durch die durchgängige Beteiligung der zukünftigen Benutzer bei der Entwicklung der Methode und der Berücksichtigung von wissenschaftlichen Erkenntnissen in diesem Bereich entgegnet.

Mehrsprachigkeit:

Bezug nehmend auf die Situation unseres Projektpartners stellt die Tatsache, dass die Geschäftsprozessmodelle ausschließlich in deutscher Sprache, die entsprechende Software allerdings immer in englischer Sprache verfasst werden, eine essentielle Herausforderung dar, was eine automatisierte Implementierung der Geschäftsprozess-modelle wiederum erschwert. Für die Modellierung der Geschäftsprozesse ist es aber sehr wichtig, dass keine sprachlichen Hürden die Beteiligten am Modellierungsvorgang behindern.

Anpassbarkeit der Werkzeuge:

Eine automatisierte Unterstützung des Transformationsprozesses setzt die Anpassbarkeit der von Werkzeugen zur Geschäftsprozessmodellierung zur Verfügung gestellten Notationen voraus. Diese Anpassbarkeit ist nicht bei allen diesbezüglichen Werkzeugen in ausreichendem Maße gegeben bzw. kann im Vorfeld der Grad der Anpassung nicht vollständig abgeschätzt werden.

Unterschiedliche Beschreibungsmethoden:

Ein wichtiges Thema sind die unterschiedlichen Beschreibungsmethoden der Beteiligten am jeweiligen Softwareentwicklungsprozess. So dokumentieren die einzelnen Fachkräfte im Zuge der Anforderungsanalyse die einzelnen Requirements vorwiegend in textueller Form, zum Beispiel in Form eines Lastenhefts (siehe Abbildung 2). Softwareentwickler verwenden hingegen unterstützend oftmals grafische Notationen (UML,…) während des Entwicklungsprozesses. Folglich bilden die unterschiedlichen Beschreibungssprachen und der dadurch notwendige Konvertierungsprozess eine weitere Erschwernis in diesem Bereich.

Definition einzelner Services[7]:

Aufgrund der unterschiedlichen Denkweisen von Geschäftsprozessmodellierern und Softwareentwicklern stellt es oft eine Schwierigkeit dar, die Kluft zwischen Betriebswirtschaft und Softwareentwicklung erfolgreich zu überbrücken. Aus diesem Grund ist es besonders wichtig, dass Entscheidungen auf der Geschäfts-prozessmodellierungsebene dokumentiert und nachvollziehbar sind, damit der Softwareentwickler den Geschäftsprozess reproduzieren und implementieren kann. Das oft unterschiedliche Verständnis ein und desselben Sachverhalts und die Tatsache, dass sich der inhaltliche und fachliche Zusammenhang einzelner Dienste und Funktionalitäten oft nicht eindeutig in Services auf der Softwareebene zusammen fassen lassen, machen eine automatische Transformation unmöglich.

[7] In diesem Projekt bezeichnet ein Service ein Software-Komponente, die inhaltlich zusammengehörige Funktionen kapselt und über eine Schnittstelle anderen Services bzw. Anwendungen zur Verfügung stellt.

3 Das Konzept im Überblick

3.1 Integrationsplattform

Ziel der Arbeit am FAW war die Entwicklung eines praxistauglichen Konzepts zur Optimierung des Zusammenspiels zwischen Geschäftsprozessmanagement und Softwareentwicklung. Einen Teil des Konzepts stellt eine Integrationsplattform dar, die teilweise automatische bzw. manuelle Transformation unterstützt.

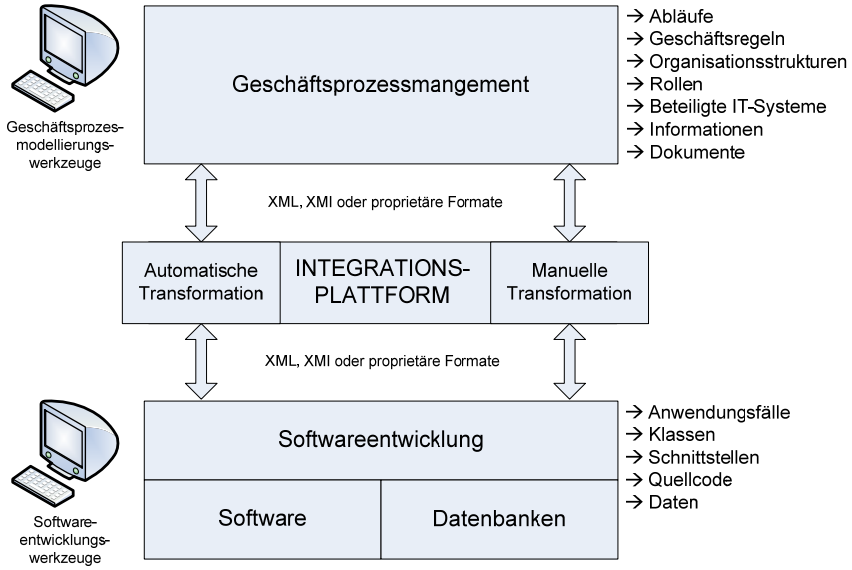


Abbildung 4: Grafische Darstellung des zu entwickelnden Konzepts

Die Integrationsplattform bietet unter anderem, folgenden Funktionen:

Datenimport

- Importieren der Adonis Daten (Adonis XML Export)

Darstellung der Daten

- Darstellung von Ablauf- und Ressourcenmodellen im Überblick in einer Baumstruktur und im Detail
- Darstellung der Services im Überblick in einer Tabelle oder im Detail mit Hilfe eines Klassendiagramms

Erstellung von Objekten

- Definieren von Services
- Zuordnung von Aktivitäten und Informationsobjekten des Geschäftsprozessmodells zu Services

Versionierung der Geschäftsprozessmodelle

- Verwaltung unterschiedlicher Modellversionen
- Kennzeichnung vorgenommener Änderungen

Informationsaustausch

- Benachrichtigung des Modellierers über Änderungen im Zuge der Integration
- Benachrichtigung des Systemarchitekten über Änderungen an den Geschäftsprozessmodellen

Export der Daten

- Exportieren der Daten aus der Integrationsplattform in ein Adonis kompatibles Format (XML)

Die Projektbeteiligten gehen davon aus, dass nur kleine Bereiche aus den Geschäftsprozessmodellen (z.B. Informationsobjekte und IT-Ressourcen, Organisationseinheiten, vgl. [HE99]) automatisch in Code bzw. UML Diagramme transformiert werden können. Die Integrationsplattform soll den Benutzer vielmehr bei der manuellen Transformation und der Definition von Services durch eine geeignete Oberfläche unterstützen.

3.2 Regelwerk

Um einen möglichst breiten Anwendungsbereich der Integrationsplattform zu gewährleisten, wurde ein Regelwerk definiert, das bei der Modellierung der Geschäftsprozesse, der Verwendung der Integrationsplattform und während der Softwareentwicklung verwendet werden soll, um einen effizienten Einsatz der Integrationsplattform zu garantieren. Das Regelwerk umfasst somit alle definierten Richtlinien aus den drei Ebenen (Geschäftsprozess, Integration und Softwareentwicklung).

4 Die Ebenen des Konzepts im Detail

Das folgende Kapitel beschreibt die einzelnen Ebenen des Konzepts, wobei insbesondre die Anforderungen aus der bisherigen Zusammenarbeit mit der Industrie berücksichtigt wurden.

4.1 Geschäftsprozessmodellierungsebene

Nachdem wie bereits erwähnt, Geschäftsprozessmodelle vorwiegend von Fachkräften aus unterschiedlichen Unternehmensbereichen erstellt werden, ohne im Speziellen auf die spätere Softwareentwicklung Rücksicht zu nehmen, ist es notwendig, den gesamten Modellierungsprozess ausführlich zu dokumentieren, damit etwaige Modellierungsentscheidungen für den Softwareentwickler nachvollziehbar sind.

Erstellung von Modellierungskonventionen:

Modellierungskonventionen legen im Rahmen der Geschäftsprozessmodellierung fest, welche Informationen in welchem Detailliertheitsgrad zu erfassen sind. Dabei werden die Modellierungsobjekte (bspw. Aktivität, Dokument, EDV-System) mit den notwendigen Attributen (bspw. Bezeichnung, Beschreibung) und die Modelltypen (bspw. Ablaufmodell, Übersichtsmodell) festgelegt. Die Modellierungskonventionen für das gegenständliche Projektziel legen den Detailliertheitsgrad der Erfassung, die Modellierungsobjekte, die notwendigen Attribute sowie die Modelltypen für eine mit der Softwareentwicklung integrierten Geschäftsprozessmodellierung fest. Diese wurden im Rahmen der Zusammenarbeit definiert und sind die Ausgangsbasis für die Anpassung bzw. Erweiterung der Modellierungskonstrukte des Geschäftsprozess-modellierungswerkzeugs sowie für die Konzeption der Schnittstellen zur Integrationsplattform.

Die Modellierungskonventionen wurden unter Berücksichtigung folgender Struktur erarbeitet und sind Rahmen eines konkreten Einsatzes entsprechend anzupassen.

- Ziele der Modellierung unter Berücksichtigung der Projektziele
- Darstellung der Modelltypen
- Definition der Modellintegration
- Definition der Modellierungsebenen
- Darstellung des Modellierungsworkflows
- Detaillierte Beschreibung der Modelltypen
- Modellierungsrichtlinien für die einzelnen Modelltypen
- Detaillierte Beschreibung der Modellierungsobjekte
- Detaillierte Beschreibung der Attribute

Erstellung von Richtlinien für die Geschäftsprozessmodellierung:

Vor der Entwicklung der eigentlichen Integrationsplattform erwies es sich als notwendig, Richtlinien zu definieren, welche bei der Erstellung der Geschäftsprozessmodelle und der darauf folgenden Transformation in Softwareapplikationen eingehalten werden müssen, um brauchbare Ergebnisse zu erzielen. Auf die dafür notwendigen Regelmäßigkeiten soll beim Durcharbeiten zahlreicher Beispielmodelle geschlossen werden. Das somit entstehende Regelwerk unterstützt den Modellierer in Kombination mit den Modellierungskonventionen bei der Erstellung der Geschäftsprozessmodelle, wobei trotz des Regelwerks der Modellierer in seiner Modellierungsfreiheit nicht zu sehr eingeschränkt werden soll.

Erweiterung der Modellierungskonstrukte des Geschäftsprozessmodellierungs-werkzeuges:

Für die Transformation der Informationen aus den Geschäftsprozessmodellen benötigt man zusätzliche Konstrukte in den Geschäftsprozessmodellierungswerkzeugen. Im Rahmen der Zusammenarbeit und der Instanzierung des Konzeptes für diesen speziellen Partner werden diese Konstrukte als Anforderungen generell festgelegt und in Werkzeugen, welche er zur Modellierung von Geschäftsprozessen einsetzt, umgesetzt.

Dokumentation der Erstellungs- und Änderungsinformation der Modelle:

Damit die einzelnen Tätigkeiten im Rahmen der Geschäftsprozessmodellierung vor allem für die Softwareentwickler nachvollziehbar sind, müssen die Informationen über die Erstellung und etwaige Änderungen der einzelnen Geschäftsprozesse hinreichend dokumentiert bzw. protokolliert werden. Aufbauend auf diese Informationen soll ein Konzept zur Versionierung der Geschäftsprozessmodelle erarbeitet werden, welches im Idealfall auch auf Rückkopplungen aus der Softwareentwicklung reagiert.

4.2 Integrationsplattform

Der Transaktionsprozess wird im Allgemeinen von einem Systemarchitekten durchgeführt. Unter Zuhilfenahme der Integrationsplattform plant der Systemarchitekt die Systemarchitektur bzw. die Softwareapplikation. Außerdem definiert er in dieser Phase die einzelnen Services und deren Kommunikation. Des Weiteren besteht die Möglichkeit, die definierten Services entsprechend den vorliegenden Geschäftsprozessen zu orchestrieren [Ro05]. Als Input stehen ihm die Informationen aus den Geschäftsprozessmodellen bzw. der Output aus dem Geschäftsprozessmodellierungswerkzeug zur Verfügung. Der Output der Integrationsplattform bzw. des Transaktionsprozesses stellt die Implementierungsinformation für den Softwareentwickler dar.

Entwicklung einer Integrationsplattform zur automatisierten und manuellen Transformation:

Zur Unterstützung des Informationsaustauschs zwischen den Werkzeugen zum Geschäftsprozessmanagement und zur Softwareentwicklung wird im Rahmen eines iterativen Softwareentwicklungsprozesses eine Integrationsplattform entwickelt. Die Integrationsplattform kommuniziert über offene Schnittstellen mit den Werkzeugen. Durch Parametrierung der Schnittstellen können Werkzeuge unterschiedlicher Hersteller angebunden werden.

Die Kernkomponente der Integrationsplattform stellt die Funktionalität zur Transformation der Objekte dar. Die Transformation ist bidirektional, es werden Geschäftsprozessobjekte in Objekte der Softwareentwicklung transformiert und umgekehrt.

Schnittstellen:

Um die angestrebte Plattformunabhängigkeit verwirklichen zu können, müssen die Schnittstellen so offen wie nur möglich gehalten werden. Hierfür eignet sich besonders

die Dokumentsbeschreibungssprache XML, da der Großteil der Werkzeuge aus den Bereichen Geschäftsprozessmodellierung und Softwareentwicklung XML Dokumente sowohl importieren, als auch exportieren kann. Im Speziellen soll die Brauchbarkeit diverser Austauschformate evaluiert werden. Als Beispiel für ein solches Austauschformat sei an dieser Stelle der Standard XMI[8] (XML Metadata Interchange) erwähnt, der oftmalige Anwendung beim Datenaustausch zwischen Softwareentwicklungswerkzeuge findet und sehr brauchbare Ergebnisse liefert. Außerdem eignet sich dieser Standard ebenfalls sehr gut für den Austausch von Diagrammen und Modellen, vor allem die Transformation von UML-Modellen betreffend.

Erstellung von Richtlinien für die Anwendung der Integrationsplattform:

Um die korrekte Anwendung der Integrationsplattform und somit einen schlüssigen Output für die Softwareentwicklung zu gewährleisten, stellt es eine Notwendigkeit dar, Richtlinien auch für den Bereich des Transformationsprozesses zu definieren.

4.3 Softwareentwicklungsebene

Der Softwareentwickler implementiert anhand der Informationen aus dem Transaktionsprozess die eigentlichen Softwareapplikationen. Wichtig ist, dass relevante Änderungen an der entwickelnden Software, die in weiterer Folge Änderungen an den Geschäftsprozessen hervorrufen können an den Geschäftsprozessmodellierer weitergegeben werden.

Erstellung von Richtlinien für die Softwareentwicklung:

Zum effizienten Umgang mit der Integrationsplattform und zur manuellen Umsetzung der Geschäftsprozessobjekte sind Richtlinien für die Softwareentwicklung zu entwickeln. Vor allem deshalb, weil der Transformationsprozess durchaus in Richtung Geschäftsprozessmodellierung vollzogen werden kann. Das heißt, dass auch Änderungen in der Softwareapplikation und damit verbundene Änderungen in den Geschäftsprozessen nur aufgrund eines strukturierten Outputs aus der Softwareentwicklung entsprechend flexibel und ohne großen Aufwand in die Geschäftsprozessebene übertragen werden können. Darüber hinaus wird in Ergänzung zur Ebene des Geschäftsprozessmanagements auch auf der Ebene der Softwareentwicklung ein Vorgehensmodell entwickelt das die Aktivitäten, Artefakte (Dokumente und Informationen die benötigt werden bzw. erstellt werden müssen) und Rollen auf dieser Ebene festlegt. Das Ziel der zu entwickelnden Methode ist die Ermöglichung eines Prozesses zur Geschäftsprozess- und Softwareentwicklung, wie im nachstehenden Kapitel ersichtlich ist.

[8] http://www.omg.org/technology/documents/formal/xmi.htm

5 Vorgehensmodell

Das Regelwerk bzw. die Integrationsplattform sollen werkzeugunterstützt das folgende Vorgehensmodell anwendbar machen.

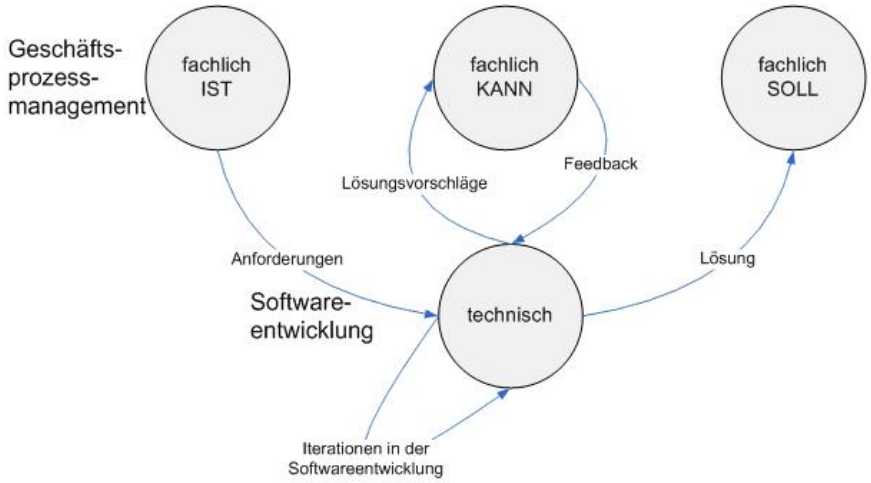


Abbildung 5: Vorgehensmodell für die Übertragung von Geschäftsprozessen in Software

Der fachliche IST-Zustand der Geschäftsprozesse in einem Unternehmen stellt die Ausgangsbasis für die Softwareentwicklung dar. Aufgrund stetiger Änderungen im Unternehmensumfeld müssen Softwaresysteme ständig den sich verändernden Geschäftsprozessen angepasst werden. Folglich beschreibt die Softwareentwicklung die technische Umsetzung der Geschäftsprozessmodelle. Während der Softwareentwicklung, vor allem in der Analyse und Designphase, können die Auswirkungen von Lösungsvorschlägen auf die Geschäftsprozesse evaluiert werden. Bereits in dieser Phase kann die Umsetzung der organisatorischen Änderungen geplant werden. Diese Auswirkungen können mit den Mitteln von Geschäftsprozessmanagementwerkzeugen analysiert werden. Die dadurch gewonnenen Erkenntnisse fließen wiederum in die Softwareentwicklung ein. Nach der Einführung des entwickelten Softwaresystems ändern sich sehr oft die zu Grunde liegenden Geschäftsprozesse, so dass zum Ende der Softwareentwicklung mit der Umsetzung der Änderungen an den Geschäftsprozessen begonnen werden kann. Die Geschäftsprozessmodelle stellen darüber hinaus eine wertvolle Ressource für die Schulung der Mitarbeiter dar. Ändern sich während der Softwareentwicklung die Geschäftsprozesse, so können auch diese Änderungen unmittelbar in die Softwareentwicklung übertragen werden.

6 Zusammenfassung und Ausblick

Im Rahmen dieses Beitrags wird ein Konzept zur Integration von Geschäftsprozessen und IT vorgestellt welches als zentralen Bestandteil eine Integrationsplattform aufweist, die die Lücke der Werkzeugunterstützung zwischen der Geschäftsprozessmodellierung und einer serviceorientierten Softwareentwicklung schließt. Das Projekt befindet sich derzeit in der Pflichtenheftphase. Workflow Management Systeme wurden noch nicht berücksichtigt und sind auch nicht Inhalt des Projekts. Die Projektbeteiligten haben allerdings die Literaturrecherche noch nicht abgeschlossen und versuchen nach wie vor alle möglichen Lösungsansätze zu sondieren.

In der weiteren Zusammenarbeit zwischen dem Institut für Anwendungsorientierte Wissensverarbeitung und dem industriellen Projektpartner soll die Praxistauglichkeit des Konzeptes weiter optimiert werden.

7 Literaturverzeichnis

[Al05] Allweyer, T.: Maßgeschneiderter Methodeneinsatz für die Entwicklung betriebswirtschaftlicher Software. In (Scheer, A.-W.; Jost, W.; Wagner, K., Hrsg.): Von Prozessmodellen zu lauffähigen Anwendungen – ARIS in der Praxis. Springer Verlag Berlin Heidelberg, 2005; S. 173-195

[MM03] Mukerji, J,; Miller, J,: MDA Guide Version 1.0.1. Object Management Group (OMG), Needham, MA 02494, USA, 2003.

[RWW03] Regner, P.; Wiesinger, Th.; Wagner, R.: Geschäftsprozessmanagement und –modellierung zur Steigerung von Effektivität und Effizienz bei der öffentlichen Ausschreibung von Softwareprojekten. In (Paul, H.; Vossen, G., Hrsg.): Tagungsband EMISA 2003: Auf dem Weg in die E.Gesellschaft. Westfälische Wilhelms-Universität Münster, Institut für Wirtschaftsinformatik, 2003, S. 72-82

[Ro05] Rohe, K.: Service-orientierte Architekturen – Strategien und Visionen. In (Kuhrmann, M.; Seifert, T.; Beneken, G.; Pizka, Dr. M.): Hot Spots der Software-Entwicklung: Service-orientierte Architekturen – Anspruch und Wirklichkeit. Technische Universität München, Institut für Informatik, Software & Systems Engineering, 2005.

[SJW05] Scheer, A.-W.; Jost, W.; Wagner, K.: Von Prozessmodellen zu lauffähigen Anwendungen – ARIS in der Praxis. Springer Verlag Berlin Heidelberg, 2005.

[He99] Heinrich, Lutz J.: Geschäftsprozessmanagement. In (Heinrich, Lutz J.):Informationsmanagement. R. Oldenburg Verlag München, 1999; S. 276 -282

Unterstützung von Kollaboration im Rahmen der Softwareentwicklung auf Basis Service-orientierter Architekturen

Thomas Karle

Forschung & Entwicklung
PROMATIS software GmbH
Pforzheimer Str. 160
76275 Ettlingen
thomas.karle@promatis.de

Andreas Oberweis

Institut AIFB
Universität Karlsruhe (TH)
Hertzstr. 16
76187 Karlsruhe
oberweis@aifb.uni-karlsruhe.de

Abstract: Im Rahmen der kollaborativen Software-Entwicklung muss die Zusammenarbeit weltweit verteilter Teams unterstützt werden. Die Realisierung von Anwendungssystemen erfolgt heutzutage vielfach auf Basis Service-orientierter Architekturen. Für die Unterstützung von Kollaboration in diesem Umfeld wird ein umfassendes formales System benötigt, das die Zusammenhänge dieser komplexen Lösungen präzise beschreibt. Der Beitrag stellt einen Petri-Netz-basierten Ansatz vor, mit dem Abläufe, Funktionen und Zusammenhänge auf verschiedenen Schichten einer Service-orientierten Architektur beschrieben werden können. Es werden insbesondere die Abläufe in Frontends und deren Zusammenspiel mit Geschäftsprozessmodellen und weiteren Schichten einer Service-orientierten Architektur betrachtet. Weiterhin wird beschrieben, wie der vorgestellte Ansatz Kollaboration in unterschiedlichen Phasen eines Software-Projekts unterstützen kann.

1 Einleitung

Viele Softwareanbieter oder Unternehmen, die sich aus strategischen Gründen für die Entwicklung firmeneigener Software entschieden haben, setzen für die Realisierung ihrer Produkte bzw. Projekte auf Offshoring [Su04]. Zunehmend werden auch Nearshore-Destinationen beispielsweise in Osteuropa attraktiv. Parallel dazu ermöglichen neue Arbeitsformen wie Telearbeit und Telekooperation, die sich durch die Verbreitung des Internets entwickelt haben, geographisch und zeitlich flexible Organisationsmodelle [BC05]. Bei der Realisierung von Systemen auf Basis moderner Service-orientierter Architekturen (SOA) sind Teammitglieder mit unterschiedlichen Kenntnissen und Fähigkeiten beteiligt. Sie realisieren Komponenten auf den verschiedenen Ebenen dieser Architekturen mit den jeweils benötigten Werkzeugen. Aufgrund der Größe der zu realisierenden Systeme, der Komplexität der Service-orientierten Architekturen und des Einsatzes moderner Entwicklungskonzepte (Extreme Programming, Aspekt-Orientierung) ist ein hohes Maß an Zusammenarbeit erforderlich. Weltweit verteilte Projektteams stellen eine weitere Herausforderung bei der Software-Entwicklung dar [CC05].

In Software-Projekten treten besondere Anforderungen hinsichtlich der Zusammenarbeit auf. Nachfolgend sind einige Beispielszenarien aufgeführt:

- Die Projektphasen Analyse, Design, Entwicklung, Test, Schulung und Inbetriebnahme werden bei Software-Projekten in der Regel von unterschiedlichen Personen durchgeführt. Eine effektive Kommunikation unter allen Beteiligten trägt maßgeblich zum Erfolg eines Projektes bei.

- Bei der Analyse müssen die fachlichen Aspekte mit den technischen Möglichkeiten für eine Umsetzung abgeglichen werden, d.h. in dieser Phase ist eine ausgeprägte Kommunikation zwischen den Teammitgliedern mit eher fachlichem Hintergrund und den eher technisch orientierten Systemarchitekten notwendig.

- Beim Design, d.h. dem detaillierten Entwurf der Software-Lösung, müssen neben den Teammitgliedern aus dem Fachbereich auch die Budget-Verantwortlichen miteingebunden werden, da die Entscheidungen, wie Umsetzungen erfolgen sollen, oft signifikante Auswirkungen auf die Kosten des Projekts haben. Die Hauptaufgabe ist hier, Transparenz für die Auswirkungen von Änderungen zu schaffen.

- Bei der agilen Software-Entwicklung, beispielsweise dem Einsatz von Extreme Programming, muss eine Umgebung bereitgestellt werden, in der eine enge Zusammenarbeit und Kommunikation innerhalb eines Teams, bis hin zum gemeinsamen Programmieren von einzelnen Funktionen unterstützt wird.

- In der Testphase bzw. im Rahmen des Betriebs von Software-Systemen müssen bei der Fehlerkorrektur ggf. frühere Teammitglieder hinzugezogen werden, um ein Problem beseitigen zu können.

Bei der Software-Entwicklung sind darüber hinaus die folgenden speziellen Rahmenbedingungen zu berücksichtigen:

- Es wird eine große Anzahl von einzelnen Komponenten erstellt, die in komplexen Gesamtsystemen zusammengefasst werden.

- Die einzelnen Komponenten können einzeln oft nicht betrieben bzw. getestet werden.

- Der Nachweis von Vollständigkeit und Korrektheit eines Gesamtsystems ist sehr aufwendig. Die Kosten für das nachträgliche Korrigieren von bereits implementierten Funktionen sind unverhältnismäßig hoch [CC05].

- Komponenten können parallel von verschiedenen Teammitgliedern bearbeitet werden.

Der Beitrag beschreibt im nachfolgenden Abschnitt zunächst den Aufbau moderner Service-orientierter Architekturen. In Abschnitt 3 werden die Grundlagen von Kollaboration erläutert. Anschließend werden in Abschnitt 4 die Mechanismen der Kollaboration im Rahmen der Entwicklung von Anwendungssystemen auf Basis Service-orientierter Architekturen untersucht. Abschnitt 5 beschreibt ein Petri-Netz-basiertes formales Modell, das als Grundlage für die Realisierung einer Umgebung für kollaborative Software-Entwicklung genutzt werden kann. Die Zusammenfassung und ein Ausblick auf weitere Schritte schließen den Beitrag.

2 Service-orientierte Architekturen

Die Realisierung von Geschäftsprozessen durch zu entwickelnde IT-Systeme erfolgt heutzutage vielfach auf Basis Service-orientierter Architekturen. Als Technik zur Implementierung von Service-orientierten Architekturen haben sich Web Services durchgesetzt [Do05]. Die für die Implementierung der Geschäftsprozesse benötigte Funktionalität wird auf mehrere Schichten verteilt [Du05]. Beispielsweise können komplexe Anwendungssysteme mit der Java Enterprise Edition (JEE) auf Basis einer solchen Architektur realisiert werden. Abbildung 1 zeigt den prinzipiellen Aufbau einer Service-orientierten Architektur und die Abhängigkeiten zwischen den Schichten.

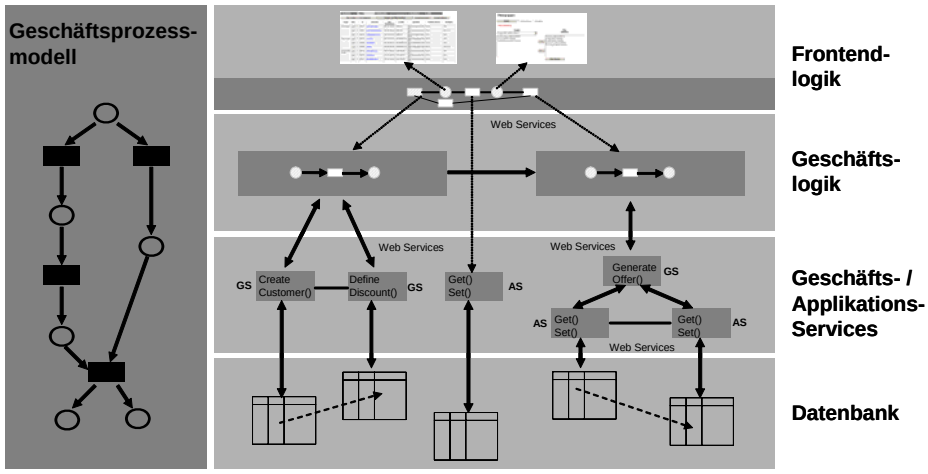


Abbildung 1: Realisierung von Geschäftsprozessen auf Basis Service-orientierter Architekturen

In der Datenbankschicht werden häufig relationale oder objekt-relationale Datenhaltungssysteme eingesetzt. Auch rein objekt-orientierte oder XML-basierte Datenhaltungssysteme sind in der Praxis zu finden. Viele relationale Datenbanken bieten inzwischen auch eine XML-basierte Sicht auf Daten, ohne dass die Daten redundant gehalten werden.

Der Zugriff auf die Daten wird in einer durchgängig Service-orientierten Architektur über die Geschäfts-/Applikations-Services-Schicht bereits mittels Web Services zur Verfügung gestellt. Die Web Services innerhalb dieser Schicht können in Applikations- (AS) und Geschäfts-Services (GS) unterschieden werden. Applikations-Services werden als wieder verwendbare Komponenten definiert, die nur innerhalb bestimmter Applikationen oder Geschäfts-Services genutzt werden können. Geschäfts-Services hingegen werden als wieder verwendbare Komponenten definiert, die direkt in unternehmensweiten oder unternehmensübergreifenden Geschäftsprozessen genutzt werden können.

Die Geschäftslogik wird einer separaten Schicht bereitgestellt. Sie greift auf die bereits Service-orientierte darunter liegende Schicht zu, welche Basisfunktionen, beispielsweise für den Zugriff auf die Datenbank oder auch komplexere Funktionen bereitstellt. In der Geschäftslogik-Schicht werden die Regeln für den Ablauf von Geschäftsprozessen implementiert. Für die Implementierung der Geschäftslogik auf Basis von Web Services hat sich die Business Process Execution Language (BPEL) etabliert. Mit BPEL können durch Orchestrierung von Web Services Geschäftsprozesse implementiert werden.

Die Steuerung des Frontends stellt die Schicht auf der Präsentationsebene bereit. Die Applikationslogik kann sich abhängig vom jeweiligen Frontend unterscheiden. Beispielsweise müssen auf einem mobilen Endgerät Eingabemasken für die Realisierung eines bestimmten Geschäftsprozesses anders dargestellt werden als auf einem PC über einen Web-Browser. Für die Steuerung der Abläufe im Frontend haben sich sogenannte Model View Controller (MVC) durchgesetzt [CST05]. Die Abläufe werden zum einen von der zugrunde liegenden Geschäftslogik und zum anderen von den Aktionen des Benutzers, beispielsweise durch Navigation über Menüs, Links oder Buttons bestimmt. Darüber hinaus muss entsprechend auf eintreffende Ereignisse reagiert werden.

3 Kollaboration

Interaktionsmechanismen, die auf dem computergestützten Austausch von Informationen basieren, werden unter dem Begriff *Computer Supported Cooperative Work* (CSCW) zusammengefasst [We96], [SDK99]. Das Forschungsgebiet CSCW beschäftigt sich mit dem Einsatz von Software zur Unterstützung von Zusammenarbeit und fasst Einflüsse aus den Forschungsgebieten Organisations- und Führungslehre, Psychologie, Informatik und Soziologie zusammen [Gesellschaft für Informatik, CSCW]. Als formale Interaktionsmechanismen mit unterschiedlichen Intensitätsgraden werden Kommunikation, Koordination, Kooperation und Kollaboration unterschieden [We96].

Kommunikation: Unter Kommunikation wird der Austausch von Informationen verstanden, wobei Informationen Daten bzw. Nachrichten in einem entsprechenden Anwendungskontext darstellen.

Koordination: Als Koordination werden Mechanismen bezeichnet, die lenkenden Einfluss auf Aktivitäten und Prozesse bei der Erstellung eines Produkts oder Erbringung einer Dienstleistung haben.

Kooperation: Als Kooperation wird die arbeitsteilige Leistungserbringung zwischen räumlich verteilten Aufgabenträgern oder Organisationen bezeichnet.

Kollaboration: Kollaboration ist ein Spezialfall der Kooperation bei der Teilaufgaben am selben Artefakt durch räumlich verteilte Aufgabenträger durchgeführt werden [Ga06], [St03].

Beispiele für Kollaboration sind die gemeinsame verteilte Erstellung oder Bearbeitung eines Dokuments, einer Software-Komponente oder der gemeinsame Entwurf eines Produkts im Rahmen der Konstruktion. Der Mechanismus der Koordination muss bei der Kollaboration um spezielle Fähigkeiten zur Sicherstellung eines konsistenten, einheitlichen und weiterverwertbaren Arbeitsergebnisses ergänzt werden. Hierzu werden, wie in Abbildung 2 dargestellt, neben dem Informations-, Wissens- und Projektmanagement, ein Konfigurationsmanagement und ein Zugriffs- und Berechtigungsmanagement benötigt. Kollaboration ist nicht auf computerbasierte Interaktionsmechanismen beschränkt. Auch andere Technologien mit denen die Zusammenarbeit von räumlich verteilten Aufgabenträgern unterstützt wird, beispielsweise das Telefon, werden mit einbezogen [KN05].

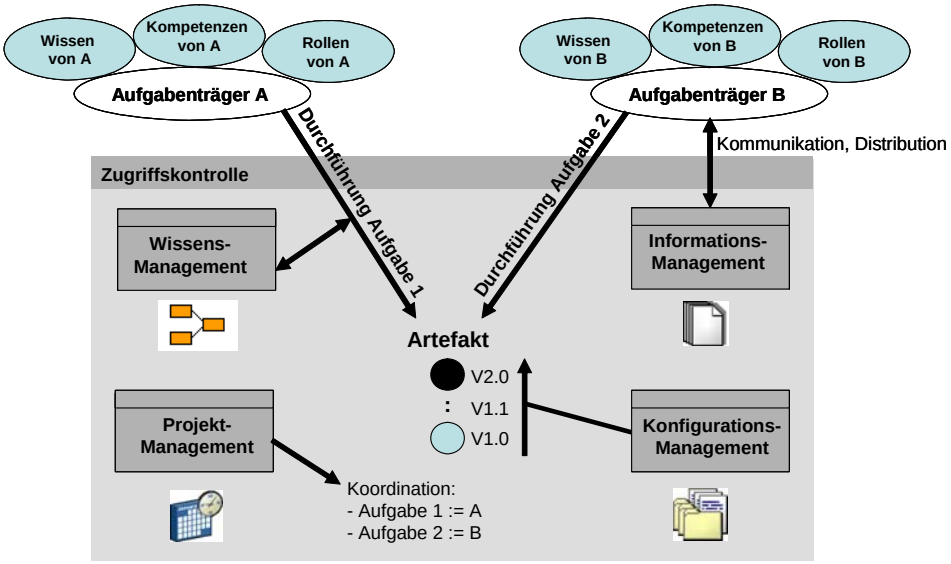


Abbildung 2: Kollaboration

Nachfolgend werden die grundlegenden Komponenten, die für den Aufbau eines Systems zur Unterstützung von Kollaboration benötigt werden, beschrieben.

Zugriffskontrolle: Der Zugang zu den Aufgaben und Informationen und die grundlegende Zugriffskontrolle für Kollaboration sollte durch ein Portal bereitgestellt werden. Portale beinhalten in der Regel bereits eine standardisierte Benutzerverwaltung, einen Login-Mechanismus und eine Zugriffskontrolle auf Ebene der Portalkomponenten, sogenannter Portlets.

Informationsmanagement: Es müssen den Beteiligten gezielt Informationen zur Verfügung gestellt werden. Für die Kommunikation können Komponenten für Mail-, Chat-, Foren, Messaging und Web-Conferencing als Bausteine in ein Kollaborations-Portal integriert werden. Darüber hinaus können über Content-Management-Komponenten sowohl Dokumente, als auch Informationen wie News über das Portal veröffentlicht werden. Durch den Einsatz von Workflow-Technologie kann der Informationsfluss zusätzlich gesteuert werden.

Projektmanagement: Über Funktionen wie Aufgabenverwaltung oder Terminverwaltung kann die Koordination erfolgen. Für die Steuerung und Überwachung kann hier ebenfalls Workflow-Technologie eingesetzt werden.

Wissensmanagement: Im Rahmen von Kollaboration wird Wissensmanagement durch eine Verknüpfung von zu erfüllenden Teilaufgaben und den dazu benötigten Informationen und Wissensträgern realisiert [FSW05]. Zu den benötigten Informationen zählen die definierten und umzusetzenden Prozesse aus der Analyse, weitere beschreibende Dokumente und die Dokumentation der benötigten Werkzeuge. Die zum Erfüllen der Teilaufgabe benötigten Werkzeuge wie beispielsweise eine CAD-Software für die gemeinsame Konstruktion eines Bauteils sollten im Rahmen des Kollaborations-Portals integriert bereitgestellt werden [GH98].

Konfigurationsmanagement: Konfigurationsmanagement wird benötigt, um den Beteiligten jederzeit einen konsistenten Zustand des aktuellen Entwicklungsstandes des gemeinsam zu realisierenden Artefakts anzuzeigen. Das Konfigurationsmanagement darf sich hierbei nicht nur auf die eigentliche Komponente beschränken, sondern muss auch beschreibende Dokumente, Modelle der Geschäftsprozesse oder weitere Informationen berücksichtigen. Der Zugriff auf eine bestimmte Version des zu realisierenden Artefakts sollte dann auch die zusätzlichen Informationen in der richtigen Version bereitstellen, beispielsweise die Dokumentation des aktuell benötigten Entwicklungswerkzeugs. Das Konfigurationsmanagement muss auch einen adäquaten Sperrmechanismus bereitstellen, der je nach Anwendungsfall unterschiedlich sein kann. Dies kann sowohl die Granularität der Sperren, als auch den Typ des Sperrmechanismus betreffen.

4 Kollaborative Software-Entwicklung

Bei der Realisierung von Systemen, basierend auf einer wie in Abschnitt 2 beschriebenen Service-orientierten Architektur, ist das Entwicklungsteam entsprechend der verschiedenen Schichten verteilt. D.h. es gibt Entwickler, die Datenbankobjekte implementieren und ggf. auch den Zugriff über entsprechende Web Services erstellen. Andere Entwickler realisieren Geschäfts-Services und Applikations-Services, die für die Implementierung der Geschäftslogik und Applikationslogik ebenfalls als Web Services bereitgestellt werden. Die Orchestrierung der Web Services zu Geschäftsprozessen mit BPEL erfolgt ggf. wieder durch andere Entwickler. Die Frontend-Programmierer implementieren beispielsweise mit Java die benötigten Applikationsbausteine und nutzen Model View Controller wie Struts oder JavaServer Faces für die Frontend-Steuerung.

Erschwerend kommt noch hinzu, dass die Teams durch Offshore-Entwicklung oder Outsourcing oft räumlich und zeitlich getrennt sind. Darüber hinaus sind zusätzlich noch die Systemanalytiker oder Mitarbeiter aus dem Fachbereich in die Realisierung involviert, beispielsweise wenn Änderungen an den Geschäftsprozessen vorgenommen werden. Aufgrund solcher Teamstrukturen werden Mechanismen für eine erfolgreiche räumlich verteilte Zusammenarbeit im Bereich Software-Entwicklung benötigt. Unter kollaborativer Software-Entwicklung versteht man die Anwendung der Konzepte und Methoden der Computer Supported Cooperative Work im Rahmen von Software-Projekten. Nachfolgend werden diese Mechanismen im Bereich der Realisierung von Anwendungssystemen auf Basis einer mehrschichtigen Service-orientierten Architektur untersucht.

5 Formales Modell

Die lose Kopplung der Schichten bringt Vorteile bzgl. der Flexibilität der einsetzbaren Technologien und der Handhabung der Flexibilität mit sich. Sie stellt jedoch eine neue Herausforderung bzgl. der Beschreibung der Abhängigkeiten zwischen den Schichten und damit des Gesamtsystems dar. Es gibt momentan kein Beschreibungsverfahren, das ein komplettes System schichtenübergreifend beschreibt und Abhängigkeiten über mehrere Schichten hinweg darstellt. Für die kollaborative Software-Entwicklung ist ein solches Beschreibungsverfahren als Basis essentiell, um beispielsweise systemgestützt Abhängigkeiten bei Änderungen erkennen zu können. Nachfolgend wird ein formales Beschreibungsverfahren erläutert, das für die kollaborative Software-Entwicklung auf Basis von Service-orientierten Architekturen dienen soll. Als einheitliche formale Beschreibungssprache werden Petri-Netze verwendet [Ba90]. Je nach Schicht werden entsprechende Varianten der Petri-Netze genutzt.

5.1 Geschäftprozessmodelle

Systemanalytiker erstellen in der Analysephase zusammen mit Mitarbeitern aus den Fachbereichen Modelle der zu realisierenden Geschäftsprozesse. Diese werden als Bestandteil von Pflichtenheften und Spezifikationen den Entwicklern bereitgestellt. Bereits in dieser Phase muss ein System zur Unterstützung von Kollaboration eine weltweit verteilte gemeinsame Erstellung der Geschäftsprozessmodelle durch entsprechendes Konfigurations-, Informations- Wissens- und Projektmanagement unterstützen. In späten Projektphasen werden auf Basis von Change Requests der Fachbereiche oftmals Prozesse geändert. Im Rahmen einer kollaborativen Software-Entwicklung sollte beides unterstützt werden. Die Änderungshäufigkeit in den späten Projektphasen kann durch Kollaboration in früheren Phasen deutlich reduziert werden [LLH06]. Die Bereitstellung der dokumentierten Prozesse aus der Analyse kann beispielsweise durch am Markt verfügbare web-fähige Content-Management-Systeme oder Wissensmanagement-Systeme erfolgen. Eine gezielte automatisierte Information von betroffenen Entwicklern über geplante oder durchzuführende Prozessänderungen in späteren Projektphasen kann damit jedoch nicht realisiert werden.

Die Voraussetzung hierzu ist eine Verknüpfung zwischen den Geschäftsprozessmodellen, deren Implementierung oder zumindest einer formalen Dokumentation der Implementierung und den beteiligten Mitarbeitern. Im formalen Modell werden zur Beschreibung der Geschäftsprozesse *Stellen/Transitions-Netze* verwendet. Abbildung 3 beschreibt einen Geschäftsprozess im CRM-Umfeld, der den Ablauf von einer Kundenanfrage bis hin zur Angebotserstellung beschreibt.

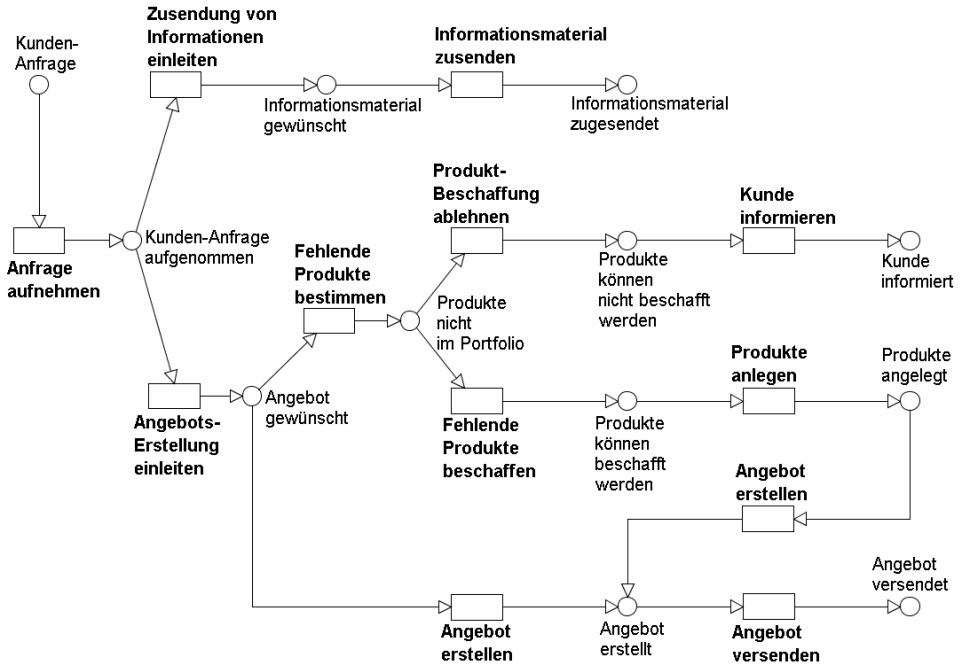


Abbildung 3: Geschäftsprozessmodell als Stellen/Transitions-Netz

5.2 Frontend-Schicht

Im Rahmen des Designs werden die Detailfunktionen und die Detailabläufe in den Applikationen, d.h. die Applikationslogik, festgelegt. In dieser Phase können neue Erkenntnisse aufgrund der Detailtiefe gewonnen werden, die zu Diskussionen und Anpassungen der bereits definierten Geschäftsprozesse aus der Analysephase führen. Sollen in späteren Projektphasen Änderungen an den darunter liegenden Web Services oder an der implementierten Geschäftslogik durchgeführt werden, so kann dies Auswirkungen auf die benötigte Funktionalität und die Abläufe in den Frontends haben. Umgekehrt können Änderungswünsche im Frontend auch Anpassungsanforderungen für Geschäftprozesse auslösen. Dieses Zusammenspiel sollte im Rahmen von kollaborativen Umgebungen unterstützt werden. Im formalen Modell wird die Applikationslogik durch so genannte *Frontend-Netze* beschrieben. Für die zuvor geschilderte Kollaboration müssen die Frontend-Netze mit den entsprechenden Geschäftsprozessmodellen und der darunter liegenden Geschäftslogik verknüpft sein.

Nachfolgend definieren wir für die Frontend-Schicht sogenannte Frontend-Netze, die den Ablauf innerhalb des Frontends beschreiben. Ein Frontend-Netz ist Tupel FN= <**P_P, P_{NE}, T_S, T_{NE}, F**> das ein Petri-Netz mit folgenden Eigenschaften darstellt:

- Es gibt zwei Typen von Stellen: P_P = Seite und P_{NE} = Navigationselement/Ereignis. P_{NE} repräsentieren beispielsweise Links, Buttons, Menüpunkte oder Ereignisse.
- Es gibt zwei Typen von Transitionen: T_S = Systemaktion und T_{NE} = Navigations-/Ereignisaktion.
- Nach einer Stelle P_P sind nur Transitionen T_{NE} erlaubt.
- Nach einer Transition T_{NE} sind nur Stellen P_{NE} erlaubt.
- Eine Transition T_{NE} ist eine Vergröberung eines Petri-Netzes, das ein exklusives Oder abbildet, d.h. sie stellt eine Entscheidung des Anwenders bzw. das Eintreffen eines bestimmten Ereignisses dar.
- P_P enthält mindestens eine XSL-Datei, in dem die grafischen Attribute der Seite beschrieben sind. Darüber hinaus enthält P_P mindestens eine XML-Datei, in dem die Struktur der Inhalte, definiert ist.
- P_{NE} enthält mindestens eine XML-Datei, das den Input für die nachfolgende Aktion enthält.
- T_S schließt eine Seite bzw. startet eine weitere Seite oder startet die darunter liegende Geschäftslogik-Schicht. Der Zugriff auf die Geschäftslogik entspricht dem Aufruf eines Geschäfts-Services.

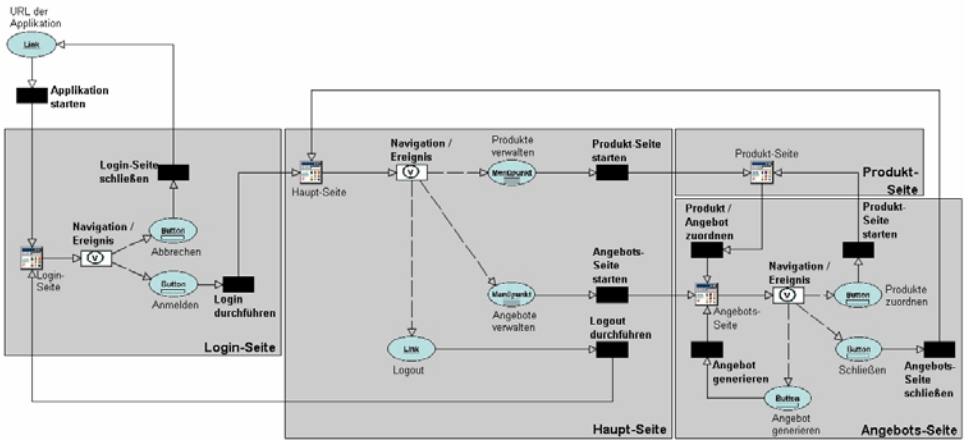


Abbildung 4: Applikationslogik als Frontend-Netz

Abbildung 4 zeigt ein Frontend-Netz, bei dem der Ablauf im Frontend einer web-basierten Applikation zur Unterstützung des oben aufgeführten Geschäftsprozesses modelliert ist. Die schwarzen Transitionen stellen Systemaktionen T_S dar, während die mit einem V markierten weißen Transitionen Navigations-/Ereignisaktionen T_{NE} repräsentieren. Die ovalen Stellen sind Navigationselemente bzw. Ereignisse vom Typ P_{NE}, während die als Icon dargestellten Seiten Stellen vom Typ P_P repräsentieren.

Zunächst wird vom Aufruf über eine URL die Applikation mit einer Login-Seite gestartet. Nach erfolgreichem Login wird die Hauptseite der Applikation gestartet. Über eine Menüstruktur kann der Anwender nun entweder die Produktseite starten, um zu prüfen ob sich ein bestimmtes Produkt im System befindet oder er kann die Angebots-Seite starten, um ein Angebot zu erstellen. Darüber hinaus kann sich der Anwender von der Hauptseite aus ausloggen. Wählt er die Angebots-Seite, dann kann er über den Aufruf der Produkt-Seite, die in diesem Fall zur Auswahl eines Produkts geöffnet wird, ein Produkt für das Angebot auswählen und zuordnen. Über den Button *Angebot generieren* wird ein darunter liegender Geschäfts-Service aufgerufen, der mit den übergebenen Parametern ein Angebot erstellt, das als XML-Datei zurückgegeben und auf der Angebots-Seite angezeigt wird. Dem Web Service werden beispielsweise ein Datum und die gewünschten Produkte mit der jeweiligen Anzahl übergeben. Mit diesen Parametern erzeugt der Web Service dann ein Angebot mit gültiger Angebotsnummer, Angebotsdatum, Gültigkeitsdatum, Produkten mit Preisen, einem Gesamtpreis, der Steuer und einem ausgewiesenen Rabatt basierend auf den Konditionen des jeweiligen Kunden.

5.3 Geschäfts- und Applikations-Services-Schicht

Im Rahmen des Designs werden die benötigten Funktionsbausteine definiert. Diese Funktionsbausteine sollen später implementiert, als Web Services zur Verfügung gestellt und in Applikationen als Applikations-Services oder in der Geschäftslogik als Geschäfts-Services verwendet werden. Änderungen der Web Services der Geschäfts- und Applikations-Services-Schicht können Auswirkungen auf die mit BPEL erstellte Geschäftslogik haben oder ziehen durchzuführende Anpassungen der Applikationslogik im Frontend oder auch der Datenstruktur in der Datenbank nach sich. Im formalen Modell werden die einzelnen Web Services der Geschäfts- und Applikations-Services-Schicht durch *Web-Service-Netze* [KM05] beschrieben. Abbildung 5 zeigt ein Beispiel für ein Web-Service-Netz, das einen Geschäfts-Service *WS_Generiere_Angebot* beschreibt, d.h. einen Web Service, der in der Geschäftslogik-Schicht genutzt werden kann. Der Web Service stellt eine Funktion bereit, die auf Basis einer XML-Datei, die ein Angebotsdatum, einen Kunden für den das Angebot erstellt werden soll und die für das Angebot ausgewählten Produkte enthält, ein entsprechendes Angebot generiert. Als Ergebnis werden die Angebotsdaten dann als XML-Datei bereitgestellt. Der Detailablauf innerhalb eines Web Services wird im formalen Modell durch ein XML-Netz beschrieben [LO03].

Die einzelnen Transitionen entsprechen dann entweder direkt Routinen, die in den im Projekt verwendeten Programmiersprachen, beispielsweise Java, implementiert sind oder noch implementiert werden sollen oder wiederum einfacheren Web Services, die hier intern verwendet werden.

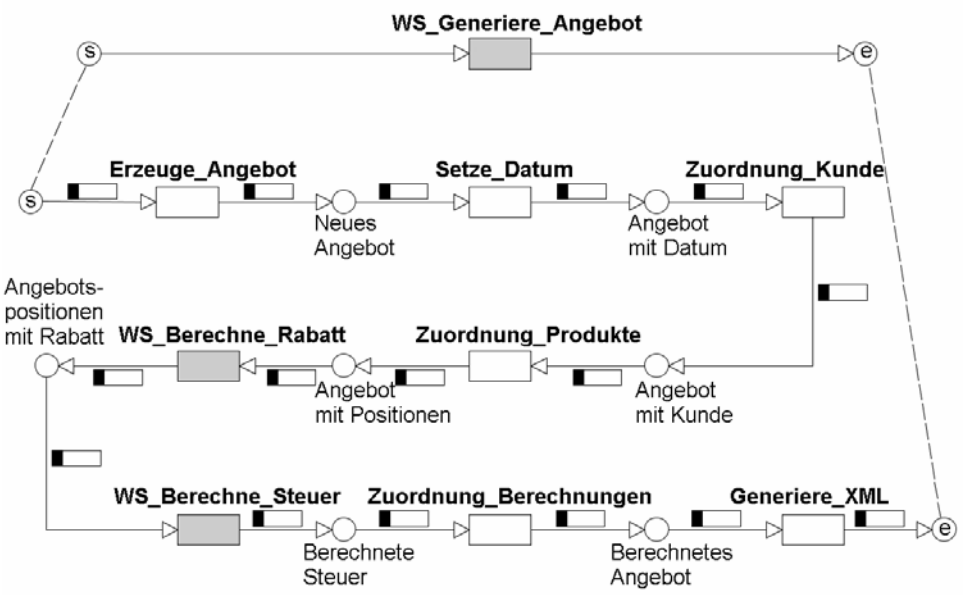


Abbildung 5: Web-Service-Netz mit Detailablauf als XML-Netz

5.4 Geschäftslogik-Schicht

Im Rahmen der Implementierung und des Betriebs wird Geschäftslogik eingerichtet und angepasst. Dies geschieht durch Orchestrierung der Web Services durch BPEL zu Geschäftsprozessen. Änderungen der Geschäftslogik haben direkte Auswirkungen auf die Geschäftsprozesse und dadurch auch auf deren Dokumentation, d.h. die Geschäftsprozessmodelle. Darüber hinaus können Änderungsanforderungen bzgl. der Geschäftslogik auch die durch Web Services realisierte Basisfunktionalität betreffen. Die durch BPEL bereitgestellte Geschäftslogik wird im formalen Modell durch die Verknüpfung von Web-Service-Netzen beschrieben [KM05].

5.5 Zusammenhänge zwischen den Schichten

Der Vorteil dieses komplett auf Petri-Netzen basierenden Ansatzes liegt in der Möglichkeit einer auf der gleichen Modellierungsmethode beruhenden Beschreibung der Zusammenhänge zwischen den Schichten. Nachfolgend werden die Zusammenhänge zwischen den verschiedenen Schichten und deren Beschreibung im formalen Modell dargestellt.

Geschäftsprozessmodell / Frontend: Eine systemtechnisch umgesetzte Transition eines Geschäftsprozessmodells wird einem oder mehreren Frontend-Netzen bzw. einem oder mehreren zusammenhängenden Teilbereichen von Frontend-Netzen zugeordnet. Dadurch wird die Beschreibung des Geschäftsprozesses direkt mit der formalen Beschreibung der Detail-Funktionalität und der Abläufe in einem entsprechenden Frontend verknüpft.

Geschäftsprozessmodell / Geschäftslogik: Ein systemtechnisch durch BPEL umgesetzter Bereich eines Geschäftsprozessmodells wird einem Bereich mit verknüpften Web-Service-Netzen der Geschäftslogik-Schicht zugeordnet.

Frontend / Geschäfts- und Applikations-Services: Eine Transition T_S (Systemaktion) eines Frontend-Netzes wird genau einem Web-Service-Netz der Geschäfts- und Applikations-Service-Schicht zugeordnet.

Geschäftslogik / Geschäfts- und Applikations-Services: Innerhalb der Geschäftslogik-Schicht werden die verfügbaren Geschäfts-Services zu implementierter Geschäftslogik orchestriert. Die einzelnen Geschäfts-Services einer mit BPEL umgesetzten oder umzusetzenden Geschäftslogik sind direkt in der Geschäfts-Logik-Schicht entsprechend der mit BPEL möglichen Ablauflogik (Sequenz, Entscheidung, Parallelverarbeitung etc.) angeordnet.

5.6 Kollaboration auf Basis des formalen Modells

Wenn im formalen Modell den jeweiligen Komponenten, d.h. den Geschäftsprozessmodellen, den Datenbankobjekten, den Applikations-Services, den Geschäfts-Services, den Abläufen in der Geschäftslogik und den Abläufen in den Frontends Mitarbeiter oder Mitarbeitergruppen zugeordnet werden, dann kann dies im Rahmen der kollaborativen Software-Entwicklung genutzt werden.

Basis für eine kollaborative Realisierung eines Anwendungssystems auf Basis Service-orientierter Architekturen ist ein feingranulares Konfigurations- und Zugriffsmanagement. Das beschriebene Modell kann als Grundlage für das Zugriffsmanagement und Steuerung von Änderungsprozessen dienen. Wird beispielsweise ein Objekt in der Datenbank-Schicht geändert oder gelöscht, dann kann diese Information automatisiert an einen Frontend-Entwickler gesendet werden, der aktuell eine Maske entwickelt, in der ein Web Service genutzt wird, der genau dieses persistente Objekt verwendet. Darüber hinaus können über diese Abhängigkeiten Sperren für den Zugriffschutz gesetzt werden. Hier sind sowohl optimistische als auch pessimistische Verfahren möglich. Bei einem pessimistischen Verfahren können bei einer Bearbeitung der persistenten Objekte, die das Objekt nutzenden Web Services für eine Bearbeitung anderer Entwickler gesperrt werden. Erst nach Freigabe des Objekts im Persistenz-Framework können die davon abhängigen Web-Services-Implementierungen wieder geändert werden. Bei einem optimistischen Verfahren lässt man bewusst Änderungen auf den verschiedenen Ebenen zu und prüft beim jeweiligen Zurückschreiben auf Konflikte.

Bei Änderungen und Erweiterungen können Zusammenhänge schnell erkannt werden. Durch die Zusammenhänge können Ansprechpartner für Diskussionen gefunden werden oder automatisch Informationen an entsprechende Mitarbeiter geschickt werden, wenn Komponenten erweitert, geändert oder gelöscht werden. Informationen können gezielt an den Komponenten innerhalb des Modells, für die der jeweilige Mitarbeiter verantwortlich ist, bereitgestellt werden. Das Modell kann als Basis für die Navigation zu den zu realisierenden Komponenten genutzt werden. Dem Benutzer können über einen solchen grafischen Zugang bereits Informationen über Sperren, durchgeführte Änderungen, verbundene Komponenten anderer Schichten, Bemerkungen anderer Teammitglieder, Termine oder weitere relevante Informationen angezeigt werden.

Ebenfalls auf der Basis des zuvor beschriebenen gemeinsamen Modells können Wissensmanagement-Strukturen im Rahmen der Realisierung implementiert werden. Die Verknüpfungen zwischen der in der Designphase erstellten Systemstruktur, in der die zu realisierenden Komponenten aufgeführt sind und den in der Analysephase definierten Geschäftsprozessen liefern den Entwicklern direkten Zugriff auf die fachlichen Informationen.

Im Bereich Projektmanagement können basierend auf dem formalen Modell Informationen über den Fertigstellungsgrad der Implementierung von Geschäftsprozessen automatisch ermittelt und angezeigt werden. Dies kann über die Zusammenhänge im formalen Modell aufgrund der Fertigstellungsgrade der einzelnen Komponenten auf Geschäftsprozessebene aggregiert berechnet werden. Die Koordination kann teilautomatisiert erfolgen, beispielsweise durch die Generierung einer Aufgabe für den Frontend-Entwickler aufgrund einer Änderung an einem Applikations-Service.

6 Zusammenfassung und Ausblick

Der Beitrag zeigt die besonderen Anforderungen bzgl. Kollaboration im Rahmen der Entwicklung von Anwendungssystemen auf Basis Service-orientierter Architekturen. Als Basis für die Unterstützung von Kollaboration in diesem Umfeld wird ein umfassendes formales System benötigt, das die Zusammenhänge dieser komplexen Lösungen beschreibt. Es wurde hier ein Petri-Netz-basierter formaler Ansatz vorgestellt, mit dem auch die Abhängigkeiten zwischen den Schichten beschrieben werden können. Aufbauend auf den beschriebenen Konzepten können Systeme zur Unterstützung von Kollaboration für die Software-Entwicklung realisiert werden. In solchen Systemen muss als Basis ein Repository mit einem entsprechenden Konfigurations- und Zugriffsmanagement für die zu verwaltenden Komponenten, Modelle, Dokumente und sonstigen Informationen realisiert werden. Darauf aufbauend können Funktionen für Kommunikation und Koordination aufgebaut werden, die aufgrund der im formalen System hinterlegten Zusammenhänge gesteuert und teilweise automatisiert werden können.

Literaturverzeichnis

[Ba90] Baumgarten B.: Petri-Netze – Grundlagen und Anwendungen, BI-Wissenschaftsverlag, Mannheim – Wien – Zürich, 1990.

[BC05] Ben E., Claus R.: Offshoring in der deutschen IT Branche – Eine neue Herausforderung für die Informatik, Informatik Spektrum, Band 28, Heft 1, Februar 2005, S. 34-39.

[CC05] Cook C., Churcher N.: Modelling and Measuring Collaborative Software Engineering, University of Canterbury, Christchurch, New Zealand, 2005.

[CST05] Cardone R., Soroker D., Tiwari A.: Using XForms to Simplify Web Programming, IBM Watson Research Center, New York, 2005.

[Do05] Dostal W., Jeckle M., Melzer I., Zengler B.: Service-orientierte Architekturen mit Web Services – Konzepte, Standards, Praxis, Spektrum Akademischer Verlag, München, 2005.

[Du05] Doubrovski V, Grundler J., Hogg K., Zimmermann O.: Service-Oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned, OOPSLA 05, San Diego, 2005.

[FSW05] Fuchs-Kittowski F., Stahn P., Walter R.: Wissensmanagement und E-Collaboration - Ein Framework für Communities, Teams und Netze zur Unterstützung kooperativer Wissensarbeit, Fraunhofer Institut für Software- und Systemtechnik ISST, Berlin, 2003.

[Ga06] Gasson S.: Boundary Knowledge-Sharing in E-Collaboration, Proc. 38th Hawaii Intl. Conf. on System Sciences, Hawaii, 2005.

[GH98] Grundy J., Hosking J.: Serendipity: Integrated Environment Support for Process Modelling, Enactment and Work Coordination. In: Automated Software Engineering, Kluver Academic Publishers, Netherlands, 1998, S. 27 – 60.

[KM05] Koschmider A., Mevius M.: A Petri Net based Approach for Process Model Driven Deduction of BPEL Code, OTM Confederated International Conferences, Agia Napa, Cyprus, 2005.

[KN05] Kock N., Nosek J.,: Expanding the Boundaries of E-Collaboration, IEEE Transactions on Professional Communication, Vol. 48, No. 1, March 2005

[LO03] Lenz K., Oberweis A.: Interorganizational Business Process Management with XML Nets, in: H. Ehrig, W. Reisig, G. Rozenberg, H. Weber (Hrsg.): Petri Net Technology for Communication Based Systems. LNCS 2472, Springer-Verlag, 2003.

[LLH06] Lefebvre E., Lefebvre L., Hen G.,: Cross-Border E-Collaboration for New Product Development in the Automotive Industry, Proc. 39th Hawaii Intl. Conf. on System Sciences, Hawaii, 2006.

[SDK99] Swaby M., Dew P., Kearney P: Model-based Construction of collaborative systems, in: BI Technology Journal, Vol 17, No 4, 1999, S. 78 – 90.

[St03] Stoller-Schai D.: E-Collaboration: Die Gestaltung internetgestützter kollaborativer Handlungsfelder, Dissertation, Universität St. Gallen, Difo-Druck GmbH, 2003.

[Su04] Sury U.: Offshoring: Die rechtlichen Dimensionen, Informatik Spektrum, Forum Offshoring, Band 27, Heft 4, August 2004, S. 365 – 395.

[We96] Wendel T.: Computerunterstützte Teamarbeit – Konzeption und Realisierung eines Teamarbeitssystems, Dissertation, Institut AIFB, Universität Karlsruhe, Deutscher Universitätsverlag, 1996.

Prozessorientierte Komposition von Diensten in der Doktorandenausbildung

Stefanie Betz, Stefan Klink, Yu Li,
Andreas Oberweis, Daniel Ried, Ralf Trunko

Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB)
Universität Karlsruhe (TH)
Hertzstr. 16
76187 Karlsruhe
{betz, klink, li, oberweis, ried, trunko}@aifb.uni-karlsruhe.de

Abstract: Aufgrund der mit dem Bologna-Prozess einhergehenden Hochschulreformen stehen die europäischen Hochschulen vor neuen Herausforderungen bei der Anpassung ihrer Organisations- und Bildungsstruktur an nationale und internationale Vorgaben und Standards. Daraus entstehen neue Anforderungen an die Prozesse im Bereich der Ausbildung von Studierenden und Mitarbeitern. Die logische Konsequenz dieser Entwicklung ist eine prozessorientierte IT-Unterstützung der Doktoranden und Habilitanden bei der Ausübung von Lehr-, Lern- und Forschungstätigkeiten, sowie den damit verbundenen organisatorischen Aufgaben. In diesem Beitrag wird ein Ansatz für eine durchgängige Prozessunterstützung während der Doktorandenausbildung, sowie ein Konzept zur technischen Realisierung vorgestellt.

1 Einleitung

Die Wandlung der Gesellschaft zur so genannten *Informationsgesellschaft* betont die Wichtigkeit der akademischen Ausbildung für Wirtschaft und Gesellschaft. Aus diesem Grund wurden mehrere Initiativen auf nationaler und europäischer Ebene gestartet, um die Ausbildung an Schulen und Universitäten sowie die berufliche Bildung zu unterstützen und zu verbessern. Die Absicht, einen europäischen Bildungsraum zu schaffen, basiert auf der so genannten *Bologna-Erklärung* [EU99]. Die Vorbereitung und Umsetzung dieser Erklärung wird *Bologna-Prozess* genannt. Ziel des Bologna-Prozesses ist es, einen europäischen Bildungsraum durch Harmonisierung akademischer Bildungsabschlüsse und Qualitätssicherungsstandards in ganz Europa zu schaffen. 2003 wurde die Integration der Doktorandenausbildung in den Bologna-Prozess beschlossen [EU03]. Ziel ist hierbei die Verzahnung des europäischen Bildungsraums mit dem europäischen Forschungsraum. In diesem Zusammenhang spielt die Doktorandenausbildung eine verbindende Rolle.

Nur wenige Doktoranden haben die Möglichkeit, ihre gesamte Arbeitszeit für ihre Promotion zu verwenden. Ein großer Teil ihrer Arbeitszeit wird für andere Aufgaben verwendet, die für einen wissenschaftlichen Mitarbeiter einer Universität anfallen, wie z.B. Aufgaben in der Lehre und administrative Aufgaben. Die Zahl der Dienstleistungen, die von Doktoranden bereitgestellt werden müssen, ist oft extrem hoch aufgrund der häufig unzureichenden personellen Kapazitäten der Universitäten. Während die Integration der Doktoranden in größere Projekte und in ein größeres Team hilft, das Qualifikationsprofil zusätzlich zur Dissertation zu vervollständigen, sind Doktoranden oft mit Lehr- oder Administrationsaufgaben belastet, die geringen Bezug zum eigenen Forschungsthema haben. Dies verlängert im Allgemeinen die Promotionsdauer, ohne die Chancen auf dem Arbeitsmarkt zu verbessern [Ke05].

Obwohl auf organisatorischer Ebene umfassende Initiativen und Programme existieren, besteht jedoch nach wie vor ein Mangel an IT-basierten Systemen, die Prozesse in der kollaborativen Forschung und der Doktorandenausbildung unterstützen. Der vorliegende Beitrag beschreibt einen Ansatz für ein service-orientiertes Konzept zur Unterstützung der während einer Promotion ablaufenden Prozesse.

Der Beitrag ist wie folgt strukturiert: nach einer Beschreibung der momentanen Situation und ähnlicher Arbeiten werden Anforderungen an ein service-orientiertes Informationssystem zur Unterstützung von Prozessen während einer Promotion diskutiert. Danach führen wir Petri-Netze und XML-Netze – ein Typ der höheren Petri-Netze – für die Modellierung von Prozessen ein und diskutieren einen Anwendungsfall für die Prozessunterstützung der Doktorandenausbildung basierend auf der Orchestrierung von Web Services. Anschließend wird eine Architektur für den beschriebenen Ansatz vorgestellt. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick auf weiteren Forschungsbedarf.

2 State of the Art

Im Gebiet der Forschung und Hochschulausbildung ist die Systemlandschaft sehr breit gefächert, insbesondere aufgrund der organisatorischen Struktur von Universitäten und anderen Forschungs- oder Bildungseinrichtungen. Betrachtet man eine Universität und ihre einzelnen Abteilungen allgemein, so kann man eine Menge unterschiedlicher Informationssysteme finden, die über die Jahre unabhängig voneinander für dieselben oder spezifische Zwecke entwickelt wurden. Aus diesem Grund mangelt es typischerweise einerseits an Kompatibilität, andererseits existieren nur wenig benutzbare Schnittstellen zwischen solchen Systemen, die man in drei Grundtypen einteilen kann. Der erste Typ deckt alle speziellen Arten von ERP-Systemen für die allgemeine Administration sowie für die Verwaltung von Studierenden, Lehrveranstaltungen und Prüfungen ab. Der zweite Typ beinhaltet Systeme, die statische Informationen verwalten und sie über das Internet verfügbar machen, wie z.B. Content Management Systeme und die darunter liegenden Webserver. Systeme, die dem dritten Typ zugehörig sind, bieten personalisierten Zugang zu standardisierten Diensten, z.B. webbasierte Lernunterstützung und webgestützte Bibliotheksdienste.

Momentan zielen europäische Förderprogramme im Bildungsbereich hauptsächlich auf Studierende im Grund- und Hauptstudium (bzw. Bachelor- und Masterstudium) ab (z.B. COMENIUS, ERASMUS und TEMPUS [EU06]) oder auf berufliche Weiterbildung (z.B. LEONARDO DA VINCI [EU06]).

Das Ziel der kürzlich gegründeten Organisation europäischer Doktoranden EURODOC[1] ist es, die Arbeits- und Studienbedingungen von jungen Wissenschaftlern zu verbessern, um ihren Beitrag zur europäischen Forschung zu steigern und die Ergebnisse der europäischen Wissenschaft zu verbessern.

Im Bereich der IT-Unterstützung für den Bildungssektor gibt es Programme wie MINERVA [EU06] und eLEARNING [EU06], sowie service-orientierte Umgebungen wie XLX [VW03], LearnServe [VW03] und ISETTA [VT06], aber diese zielen hauptsächlich auf Open und Distance Learning bzw. eLearning-Konzepte ab, sowohl für die Ausbildung von Studierenden als auch im Bereich der beruflichen Weiterbildung. Es mangelt jedoch an spezifischen Programmen für den IT-Einsatz zur Unterstützung von Doktoranden.

Spezifische Unterstützung für Doktoranden bietet das Human Potential Programm der Europäischen Union[2]. Dieses Programm zielt darauf ab, Ausbildung und Mobilität von Wissenschaftlern aus nahezu allen Forschungsgebieten in ganz Europa zu unterstützen, z.B. in der MARIE CURIE Action[3], bei welcher der Austausch von Doktoranden und Postdoktoranden unterstützt wird. Aber die Unterstützung durch das Human Potential Programm bezieht sich hauptsächlich auf den organisatorischen Bereich.

Angesichts der wachsenden und unterschiedlichen Anforderungen, welche die Doktoranden erfüllen müssen, wird eine IT-basierte Unterstützung der Prozesse, die während der Promotion ablaufen, immer notwendiger.

Viele Universitäten unternehmen heute große Anstrengungen[4], um ihre Informationssysteme in ein einziges Universitätsportal zu integrieren und bieten Zugang sowohl zu bestehenden Diensten, als auch zu modernen, anspruchsvollen Diensten zur Unterstützung von Studierenden während ihres Studiums. Ein neuartiger Ansatz für eine umfassende Integration, realisiert durch eine service-orientierte Architektur, wird momentan von der Universität Karlsruhe (TH) realisiert [Ju06]. Basisfunktionalitäten von diversen Altanwendungen auf unterschiedlichen Plattformen werden in Web Services gekapselt, die dann zu komplexeren Mehrwertdiensten orchestriert werden.

Die Herausforderung ist es, Doktoranden durch das Anbieten spezieller Dienste, die sie während ihrer Promotion und für ihre Forschungsaktivitäten benötigen, zu unterstützen, indem alle Aspekte aus Forschung und Lehre durchgängig als Prozesse betrachtet und modelliert werden. Hierdurch wird es Doktoranden und Wissenschaftlern ermöglicht,

[1] http://www.eurodoc.net/
[2] http://www.cordis.lu/improving/
[3] http://cordis.europa.eu.int/mc-opportunities/
[4] In Deutschland z.B. Universität Karlsruhe (TH), die TU München, die RWTH Aachen, die Universität Hamburg und die FU Berlin.

obligatorische Prozessmodelle, die von der Universität bereit gestellt werden, zu nutzen (z.B. Modelle für Promotionsordnungen oder administrative Prozesse), zusätzlich individuelle Prozesse zu modellieren (z.B. persönliche Aufgabenlisten) und andere Personen im Sinne kollaborativer Arbeit (z.B. Koautorenschaft oder Projektmitarbeit) den Prozessen und den zugehörigen Ressourcen zuzuweisen.

3 Anforderungen

Die Aufgaben während einer Promotion umfassen das Recherchieren, das Arbeiten an der Dissertation, das Anfertigen von Publikationen und die Teilnahme an Konferenzen, das Unterrichten und Betreuen von Studierenden, die Mitarbeit in Forschungsprojekten sowie andere wissenschaftliche oder organisatorische Aufgaben (z.B. Verantwortung für Bibliotheken und Infrastruktur, Arbeit in universitären Gremien oder Konferenzkomitees etc.).

Um trotz dieser vielen Aufgaben die Produktivität zu erhöhen und ein schnelleres und zielgerichteteres Arbeiten an der Dissertation zu ermöglichen, ist eine spezifische IT-basierte Unterstützung für Doktoranden hilfreich. Abbildung 1 gibt einen exemplarischen Überblick über die Anforderungen an eine IT-Unterstützung für Doktoranden.

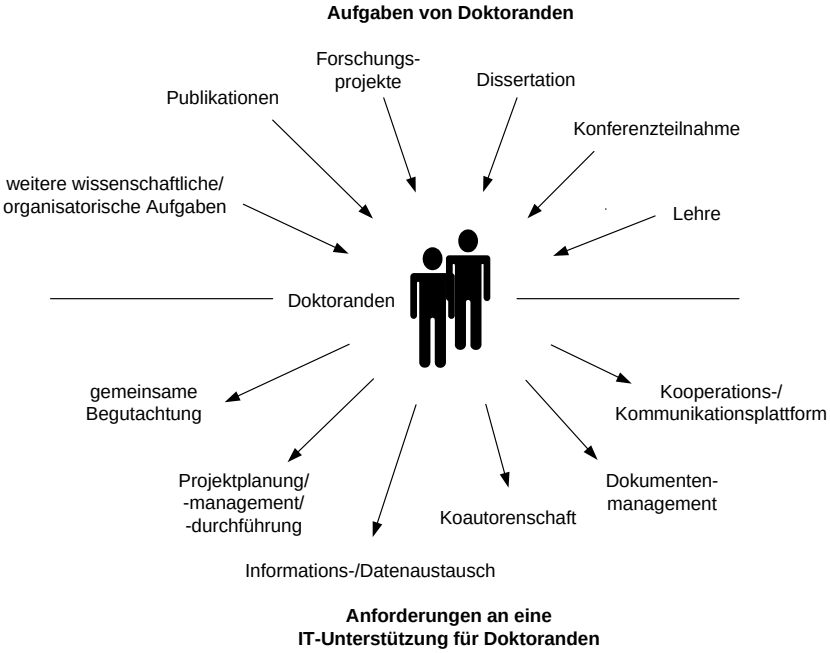


Abbildung 1: Exemplarische Übersicht der Anforderungen

Die Implementierung eines Informationssystems zur Unterstützung kollaborativer Forschung und Doktorandenausbildung muss die momentanen Promotionsgegebenheiten und fächerspezifische Besonderheiten (z.B. unterschiedliche Promotionsordnungen, unterschiedliche Publikationsrichtlinien etc.) berücksichtigen. Die notwendigen Veränderungen verlangen eine stärkere Formalisierung der Struktur der Promotionsordnungen. In diesem Zusammenhang müssen die Prozesse innerhalb einer Promotion stärker als bisher strukturiert werden.

Hinsichtlich der bereits erwähnten Aufgaben kann eine Menge von Basisfunktionalitäten identifiziert werden, die in ein Informationssystem zur Unterstützung kollaborativer Forschung und Doktorandenausbildung integriert werden müssen. Betrachtet man die Doktorandenausbildung aus einer prozessorientierten Sicht, so wird der Ausbildungsprozess durch unterschiedliche Arten von Regularien oder Richtlinien allgemein definiert, die von der Universität, ihren Fakultäten oder ihren Instituten vorgegeben werden. Mehrere obligatorische Aufgaben und Restriktionen werden von einem Doktoranden erfüllt, abhängig von seiner jeweiligen Rolle (z.B. als Autor, als Lehrender, als externer Doktorand, als Stipendiat, oder als Angestellter).

Auf oberster Ebene ist der Promotionsprozess durch eine Promotionsordnung für jede Fakultät unterschiedlich definiert. Solche Regularien spezifizieren allgemein die Menge und die Art der Arbeit, die von einem Doktoranden in einer definierten Zeitperiode erledigt werden muss, z.B. Forschungsaktivitäten, Veröffentlichung von Ergebnissen, Projektmitarbeit oder Promotionsprüfung. Zusätzlich ist jede Aufgabe mit verschiedenen administrativen Aufgaben verbunden wie z.B. der Dienstreiseantrag oder der Urlaubsantrag, die ebenso durch mehrere Regularien und Restriktionen definiert werden. Jede dieser Aufgaben wird als ein eigenständiger Teilprozess aufgefasst und modelliert, um anschließend bei der Instanziierung individuell vom Doktoranden angepasst zu werden.

Für die Unterstützung kollaborativer Arbeit muss ein Informationssystem einen verteilten Arbeitsplatz für Informations- und Datenaustausch sowie für Kooperation und Kommunikation zwischen Doktoranden und anderen Wissenschaftlern repräsentieren. Allgemeine Szenarien sind Planung, Management und Durchführung von Projekten, kollaborative Forschung, Koautorenschaft, gemeinsame Begutachtungen etc. In dieser Hinsicht muss ein Doktorand, der für einen Teilprozess (z.B. als Hauptautor einer Koautorenschaft oder als Projektleiter) verantwortlich ist, in der Lage sein, anderen Akteuren bestimmte Aufgaben zuzuweisen (z.B. die Zuweisung von Kollegen zu Aufgaben in einem internen Begutachtungsprozess). Jedem Akteur, der an einem Prozess teilnimmt, werden verschiedene Dienste angeboten. Solche Dienste sind beispielsweise Monitoring, Controlling und Reporting des gesamten Prozesses oder einzelner Aufgaben.

Die Erfüllung dieser Anforderungen bietet Doktoranden die Möglichkeit, zielgerichteter zu arbeiten. Der in diesem Beitrag präsentierte Ansatz fokussiert die Prozessunterstützung. Die Verwendung eines hierarchischen Prozessmodells und die Kapselung bestimmter Prozesse durch Dienste ermöglicht das Angebot von höherwertigen und wieder verwendbaren Diensten.

4 Prozessmodellierung mit Petri-Netzen

Petri-Netze [Re86, RR98] sind eine formale Beschreibungssprache mit einer grafischen Repräsentation und einer mathematischen Fundierung. Aufgrund dieser Eigenschaften werden sie häufig als Prozessmodellierungssprache verwendet. Im Gegensatz zu anderen grafischen Notationen ermöglichen sie die Simulation von Prozessmodellen und die Analyse des Verhaltens. Notationen wie Ereignisgesteuerte Prozessketten (EPK) [SN00], die in praktischen Anwendungen für die Geschäftsprozessmodellierung weit verbreitet sind, können als semi-formale Varianten von Petri-Netzen mit einer spezifischen grafischen Schnittstelle betrachtet werden.

Ein Petri-Netz ist ein gerichteter bipartiter Graph, der aus einer Menge von statischen Elementen, *Stellen* genannt, und einer Menge von dynamischen Elementen, *Transitionen* genannt, besteht; Stellen und Transitionen werden durch gerichtete Kanten verbunden. Ein Petri-Netz wird formal durch das Tripel $N = (S, T, F)$ definiert. S ist die Menge der Stellen, T ist die zu S disjunkte Menge der Transitionen und $F \subseteq (S \times T) \cup (T \times S)$ ist eine *Flussrelation*, welche die Menge der Kanten bezeichnet. Die Elemente aus S werden grafisch repräsentiert durch Kreise, Elemente aus T durch Rechtecke und Elemente aus F als gerichtete Kanten zwischen Stellen und Transitionen.

Eine Stelle $s \in S$ ist Eingabestelle einer Transition $t \in T$, wenn eine gerichtete Kante von s nach t existiert. Eine Stelle s ist Ausgabestelle einer Transition t, wenn eine gerichtete Kante von t nach s existiert. Die Menge aller Eingabestellen einer Transition t wird mit $\bullet t$ bezeichnet und *Vorbereich* genannt. Die Menge aller Ausgabestellen einer Transition t wird mit $t\bullet$ bezeichnet und *Nachbereich* genannt [Re86].

Eine Stelle eines Petri-Netzes wird interpretiert als Container für Datenobjekte, die als *Marken* bezeichnet werden. Die Anzahl der Marken einer Stelle wird durch die *Markierung* dieser Stelle repräsentiert. Für die Modellierung von Prozessen auf unterschiedlichen Abstraktionsebenen wurden zwei Klassen von Petri-Netzen vorgeschlagen, die eine Vielzahl von Varianten abdecken, welche unterschiedliche Interpretationen einer Markierung zulassen. In der Klasse der einfachen Petri-Netze werden Boolesche Datenobjekte durch genau eine Marke (Bedingungs-/Ereignis-Netze) bzw. durch eine Multimenge von nicht unterscheidbaren Marken (Stellen-/Transitionen-Netze) repräsentiert [Re86]. In der Klasse der höheren Petri-Netze (z.B. Gefärbte Petri-Netze [Je93], Prädikate-/Transitionen-Netze [GL81], XML-Netze [LO03]) repräsentieren die Marken Datenobjekten, die einem bestimmten Datentyp entsprechen.

In jeder Variante von Petri-Netzen kontrollieren Transitionen den Fluss von Datenobjekten, die aus den Stellen ihres Vorbereichs entnommen und den Stellen ihres Nachbereichs zugewiesen werden. Transitionen können mit einem logischen Ausdruck (Prädikatenlogik der ersten Stufe) beschriftet sein, bestehend aus einer Bedingung und einer Operation für den Verbrauch, die Produktion oder Manipulation der Datenobjekte.

In einem XML-Netz [LO03] wird jede Stelle durch ein XML Schema-Diagramm beschriftet, wobei das XML Schema den so genannten *Stellentyp* repräsentiert. Stellen werden daher als Container für XML-Dokumente, welche dem jeweiligen Stellentyp ge-

nügen, interpretiert. XML-Dokumente stellen hier die prozessrelevanten Datenobjekte dar. Die Ausführung der Aktivitäten wird durch den Fluss der XML-Dokumente, d.h. das Schalten der Transitionen, definiert. Eine Transition wird mit einem prädikatenlogischen Ausdruck beschriftet, dessen freie Variablen in den Inschriften der adjazenten Kanten, den so genannten *Filterschema*-Diagrammen, enthalten sein müssen. Filterschema-Diagramme repräsentieren Lese- oder Manipulationsfilter für die jeweiligen Zugriffsarten Lesen bzw. Ändern, Erzeugen oder Einfügen und Löschen eines oder mehrere Datenobjekte. Beim Schalten einer Transition werden für eine existierende Variablenbelegung XML-Dokumente aus den Markierungen der Vorbereichsstellen gelesen oder gelöscht und in die Markierungen der Nachbereichsstellen eingefügt bzw. bereits existierende XML-Dokumente werden geändert. Ein Beispiel für ein beschriftetes XML-Netz zeigt Abbildung 4 im folgenden Abschnitt "Anwendungsfall".

5 Anwendungsfall

Ein typischer Prozess im Rahmen einer Promotion ist die Teilnahme an einer Konferenz. Die Teilnahme kann einerseits aktiv erfolgen durch Einreichen eines Konferenzbeitrags und den Vortrag einer Präsentation über das eigene Forschungsgebiet sowie das Erhalten von Kritik und Anregungen von anderen Wissenschaftlern. Andererseits kann man auch passiv als Zuhörer an einer Konferenz teilnehmen. Beide Arten der Teilnahme bieten die Möglichkeit zum Austausch wissenschaftlicher Ideen mit anderen Wissenschaftlern.

Für diesen Beitrag wurde der Prozess "Konferenzteilnahme" als Anwendungsfall ausgewählt, um den präsentierten Ansatz zu illustrieren. Abbildung 2 beschreibt den genannten Anwendungsfall auf vereinfachte Weise als einfaches Petri-Netz. In diesem Anwendungsfall entscheidet sich ein Doktorand für eine aktive Teilnahme an einer Konferenz. Hierdurch wird ein komplexer Teilnahme-Prozess angestoßen, der als Komposition aus mehreren Teilprozessen gesehen werden kann. Hierzu gehören das Verfassen und Einreichen eines Konferenzbeitrags, das Beantragen einer Dienstreise, die Berücksichtigung organisatorischer Aspekte einer Konferenzteilnahme (z.B. Registrierung und Bezahlung der Konferenzgebühr), die Anfertigung einer Präsentation, die Organisation der Reise und schließlich die eigentliche Teilnahme an der Konferenz. Diese Teilprozesse können als lose gekoppelte, aber disjunkte Dienste betrachtet werden. Diese kommunizieren über einheitliche Schnittstellen miteinander und weisen spezifische Abhängigkeiten auf. Dies bedeutet, dass jeder Teilprozess als eigenständiger Prozess gehandhabt werden kann, der für andere Zwecke wieder verwendet werden kann. Der Teilprozess bzw. Dienst "Publikation einreichen" kann auch für die Anfertigung von Zeitschriftenartikeln verwendet werden, die Services "Reiseantrag einreichen" und "Reise organisieren" können für jegliche Art von Dienstreisen verwendet werden, die Services "Teilnahme organisieren" und "An Konferenz teilnehmen" können sowohl für aktive als auch für passive Teilnahme an jeder Art von Veranstaltung verwendet werden. Der Service "Präsentation anfertigen" kann verwendet werden für die Anfertigung von Präsentationen für jegliche Zwecke.

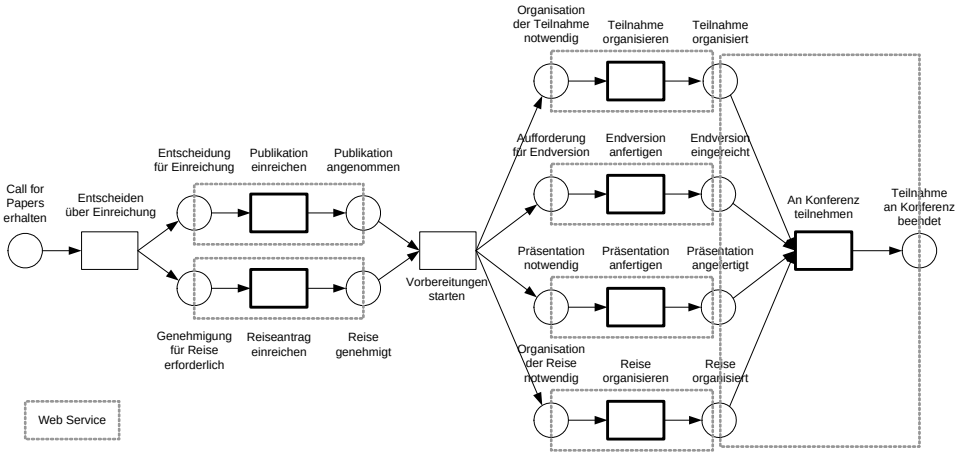


Abbildung 2: Prozess "Konferenzteilnahme"

Aus Gründen der Übersichtlichkeit sind die Teilprozesse in Abbildung 2 als aggregierte Transitionen (dick umrandet) modelliert. Im Zusammenhang mit Petri-Netzen bedeutet *Aggregation* das Ersetzen eines stellen- oder transitionsberandeten Teilnetzes durch jeweils eine einzelne Stelle oder Transition. Die Umkehrung der Aggregation wird *Verfeinerung* genannt [GL81].

Abbildung 3 zeigt die unterschiedlichen Abstraktionsebenen des Teilprozesses "Publikation einreichen". Das Teilnetz, welches der aggregierten Transition "Publikation einreichen" zugrunde liegt, zeigt zwei alternative Pfade. Diese sind abhängig davon, ob ein Abstract vor Einreichung des eigentlichen Beitrags eingereicht werden muss. In diesem Teilnetz kann z.B. die Transition "Publikation anfertigen" eine Schnittstelle für die Koordination mit den möglichen Koautoren des Beitrags besitzen. Ein Teilnetz kann weitere Teilnetze (d.h. aggregierte Transitionen) beinhalten. Im vorgestellten Beispiel repräsentiert die aggregierte Transition "Publikation mit Abstract anfertigen" ein Teilnetz auf der nächsten Abstraktionsebene, welches über weitere Schnittstellen verfügen kann, um die Anfertigung des Abstracts und des eigentlichen Konferenzbeitrags zusammen mit möglichen Koautoren zu koordinieren. Das Anfertigen von Publikationen kann auch Services wie Videokonferenzen oder Literaturrecherche in digitalen Bibliotheken beinhalten. Der Einfachheit halber zeigt das beschriebene Prozessmodell einen idealisierten Prozessverlauf (wie die Stelle "Publikation angenommen" zeigt). Üblicherweise sollten Mechanismen zur Ausnahmebehandlung in das Prozessmodell integriert werden, um Situationen wie "Ausfall eines Koautors", "Ablehnung einer eingereichten Publikation" oder "Ablehnung eines eingereichten Dienstreiseantrags" abzufangen. Außerdem beschreibt der vorgestellte Anwendungsfall eine aktive Konferenzteilnahme. Eine passive Teilnahme könnte aus den Teilprozessen "Reiseantrag einreichen", "Teilnahme organisieren", "Reise organisieren" und "An Konferenz teilnehmen" bestehen. Die technische Umsetzung des vorgestellten Ansatzes kann realisiert werden, indem jede aggregierte Transition als Web Service gekapselt wird (in Abbildung 3 angedeutet durch die grau gestrichelten Kästchen). Die Stellen des Vor- und Nachbereiches der jeweiligen Transition fungieren als Schnittstellen für die Kommunikation zwischen den verschiedenen

Web Services. Aufgrund der Tatsache, dass Web Services auf XML-Technologie basieren, sind XML-Netze sehr gut geeignet für die Prozessmodellierung im Allgemeinen [LO03] und für die Orchestrierung von Web Services [LO04] im vorgestellten Ansatz.

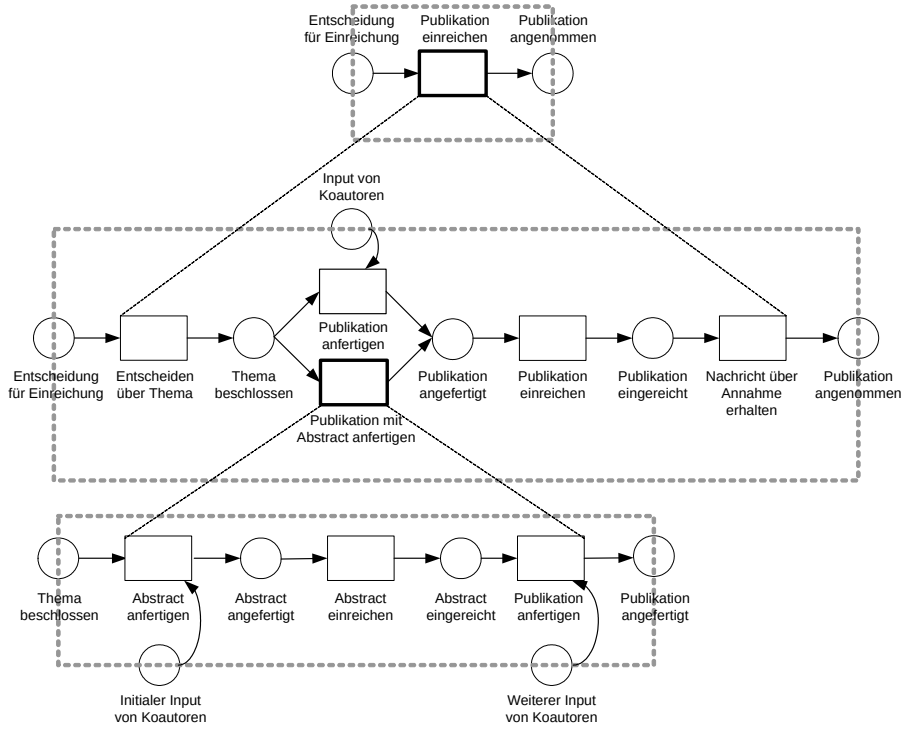


Abbildung 3: "Publikation einreichen"

Durch den vorgestellten Ansatz wird die Wiederverwendbarkeit der unterschiedlichen Teilprozesse erhöht. Außerdem ermöglicht der Ansatz eine effiziente Modellierung und Ausführung von spezifischen Prozessen, die während der Promotionszeit durchgeführt werden müssen.

Bezüglich der Unterstützung kollaborativer Forschung ist anzumerken, dass die im Anwendungsfall beschriebenen Services von mehreren Doktoranden verwendet werden können, z.B. wenn sie eine gemeinsame Publikation anfertigen. Die Abstimmungsbemühungen unter den teilnehmenden Doktoranden können als Kommunikation zwischen den jeweiligen individuellen Publikationsprozessen (gekapselt in Web Services) betrachtet werden. Weitere mögliche Anwendungsfälle sind z.B. die Organisation von Doktorandenseminaren oder die gemeinsame Arbeit an einem Forschungsthema.

Abbildung 4 zeigt ein einfaches XML-Netz für den Teilprozess "Publikationseinreichung" mit XML Schema- und Filterschema-Diagrammen, die jeweils den Stellen und Kanten zugeordnet sind. Das XML-Wurzelelement "Publikation" besteht aus den Kindelementen "Titel", "Autor", "Referenz" und "Angenommen", wobei das Element "Referenz" auf die eigentliche Publikation verweist. Das Element "Angenommen" ist vom Typ

Boolean und beschreibt den Annahmestatus der Publikation. Beim Schalten der Transition "Publikation einreichen" werden die XML-Dokumente, welche die Metadaten von Publikationen enthalten, also einen Titel, einen oder mehrere Autoren und ein Element "Angenommen" mit dem Wert "false" besitzen, aus der Stelle "Entscheidung für Einreichung" gemäß dem zur ausgehenden Kante gehörenden Filterschema gelöscht. Entsprechend dem Filterschema, das zur aus der Transition "Publikation einreichen" ausgehenden Kante gehört, und dem XML Schema der Stelle "Publikation angenommen" werden XML-Dokumente erzeugt, wobei der Wert des Elements "Angenommen" auf "true" gesetzt wird. Die schwarzen Balken in den Filterschema-Diagrammen repräsentieren *Manipulationsfilter*, die zum Erzeugen oder Löschen von Dokumenten eingesetzt werden. Die Rechtecke mit einem stilisierten "A" stellen jeweils einen *Elementplatzhalter* dar, der den Datentyp *AnyType* der XML Schema Definition (XSD) [FW04] besitzt und den allgemeinsten Elementtyp darstellt. In einer Instanz dieses Datentyps können an Stelle dieses Elementplatzhalters Elemente beliebigen Typs stehen.

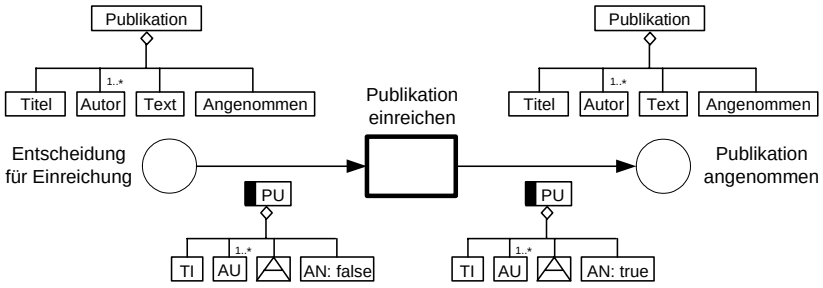


Abbildung 4: XML-Netz "Publikation einreichen"

6 Architekturkonzept

Die Integration von Informationssystemen innerhalb einer heterogenen IT-Infrastruktur ist nach wie vor ein wichtiger Aspekt der Softwareentwicklung. Jedoch existieren für Hochschulen keine umfassenden Integrationslösungen, wie z.B. kommerzielle Lösungen zur Anwendungsintegration in Unternehmen (Enterprise Application Integration, EAI). Viele dieser Lösungen basieren auf dem Konzept der *Service-orientierten Architektur* (SOA), welches eine prozessorientierte Integration von bestehenden Altsystemen (Legacy-Systeme) in einer heterogenen IT-Infrastruktur realisiert. Die Idee der SOA ist es, Zugriff auf Funktionalitäten bestehender Systeme in Form von Diensten mit einheitlichen Schnittstellen zu ermöglichen.

Die Architektur des vorgestellten Ansatzes beruht im Allgemeinen auf dem Konzept der SOA und im Besonderen auf deren Realisierung mittels Web Services. Web Services sind plattformunabhängig, verfügbar über das Internet und basieren auf gängigen und etablierten Protokollen und Standards, wie z.B. TCP/IP, HTTP, XML, SOAP, WSDL, UDDI sowie zahlreiche Erweiterungen zu Sicherheits- und Interoperabilitätsaspekten. Abbildung 5 zeigt das von [Ju06] abgeleitete vierschichtige Architekturkonzept für den vorgestellten Ansatz.

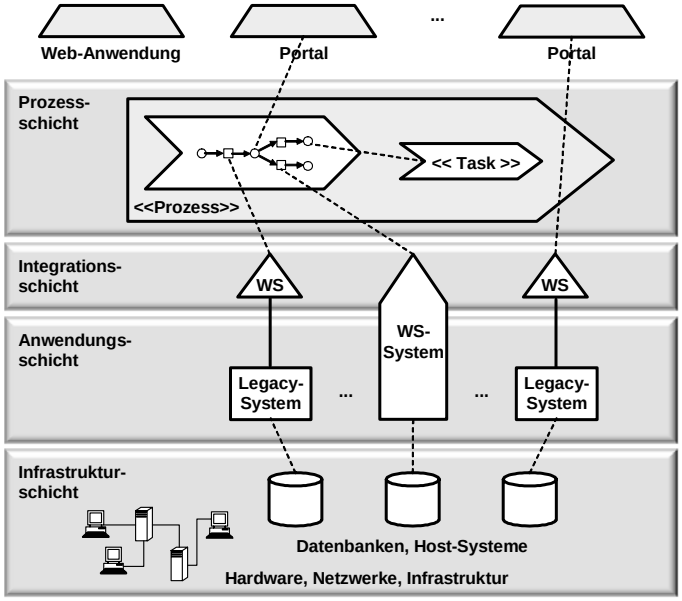


Abbildung 5: Architekturkonzept

Zur Infrastrukturschicht zählen sämtliche Hardware- und Netzwerkkomponenten, sowie Datenbank- und Host-Systeme. Zu integrierende Altsysteme und Anwendungen sowie Systeme, die bereits über Web Service-Schnittstellen verfügen, befinden sich innerhalb der Anwendungsschicht. Die Integrationsschicht beinhaltet sämtliche Web Services, die einen Zugriff auf Systeme der Anwendungsschicht ermöglichen. Innerhalb der Prozessschicht werden Web Services der Integrationsschicht mit Hilfe von XML-Netzen zu höherwertigen Diensten orchestriert (siehe Abschnitt 5). Aufgrund dessen kann der Gesamtprozess der Promotion als ein einziger durchgängiger Prozess auf der Anwendungsschicht, der sich aus mehreren Teilprozessen und Aufgaben zusammensetzt, angesehen werden. Die Benutzerinteraktion wird durch Web-Anwendungen oder Portale ermöglicht. Innerhalb eines Universitätsportals können somit einzelne Web Services der Integrationsschicht oder solche, die Teil einer Orchestrierung sind, integriert und zugänglich gemacht werden.

7 Fazit und Ausblick

Obwohl mehrere Initiativen für die Unterstützung der Ausbildung an Schulen und Universitäten existieren, herrscht nach wie vor ein substanzieller Mangel an IT-basierten Systemen. Der in diesem Beitrag vorgestellte Ansatz bietet spezielle Dienste, die während einer Promotion benötigt werden, indem er alle relevanten Aspekte aus Ausbildung und Forschung als Teilprozesse betrachtet. Für die kollaborative Forschung und Doktorandenausbildung wurde ein neuer Ansatz vorgestellt. Dieser basiert auf einer serviceorientierten Architektur, die eine prozessorientierte Integration von Informationssystemen innerhalb einer heterogenen Systemlandschaft ermöglicht.

Für die Modellierung der Prozesse wurden Petri-Netze als formale Beschreibungssprache mit einer grafischen Repräsentation und einer mathematischen Fundierung eingeführt. Sie können für die Simulation von Prozessmodellen und für die Analyse deren spezifischen Verhaltens und Charakteristika verwendet werden. Durch die Verwendung von XML-Netzen – eine Variante höherer Petri-Netze – können Stellen interpretiert werden als Container für XML-Dokumente, welche die jeweiligen prozessrelevanten Datenobjekte repräsentieren. Die Ausführung der Aktivitäten wird durch den Fluss der XML-Dokumente, d.h. das Schalten der Transitionen, definiert. XML-Netze eignen sich besonders für die Modellierung der komplexen dokumentenbasierten Prozesse, die in Forschung und Lehre auszuführen sind.

Der vorgestellte Ansatz trägt hierbei in vielerlei Hinsicht zur Unterstützung von Doktoranden bei:

- Durch die Möglichkeit einer intensiveren Beschäftigung mit dem eigenen Forschungsthema kann die Promotionsdauer verkürzt werden.

- Die Promotionsbedingungen verbessern sich im Allgemeinen durch teilweise Entlastung von administrativen Aufgaben.

- Durch die Verwendung eines service-orientierten Ansatzes verbessern sich die Kommunikation der Doktoranden untereinander und die Kommunikation zwischen Doktoranden und anderen Experten.

- Ebenso unterstützt der Ansatz das kollaborative Problemlösen bei der gemeinsamen Arbeit an Forschungsthemen, was auch der Internationalisierung der Doktorandenausbildung zugute kommt.

- Eine kollaborative Qualitätssicherung (z.B. durch Integration von externen Gutachtern in die Betreuung von Doktoranden) wird in der Doktorandenausbildung ermöglicht.

- Die Integration von digitalen Bibliotheken, Recommender-Systemen, kontextabhängigen Diensten etc. für Publikationen erleichtert die Arbeit der Doktoranden.

- Durch die Verwendung von Web Services wird die Interoperabilität in heterogenen Software- und Hardware-Umgebungen gewährleistet.

- Auch mobile Arbeitsplätze (z.B. beim Besuch einer Konferenz) werden durch den Web Service-orientierten Ansatz unterstützt.

- Die Wiederverwendbarkeit von Teilprozessen als Web Services ermöglicht eine Modularität mit geringen Redundanzen und trägt zur Flexibilität der eingesetzten IT-Systeme bei.

Da der vorgestellte Ansatz sich noch in einem frühen Forschungsstadium befindet, wurde nur das Konzept an sich beschrieben. Weitere Schritte beinhalten die Implementierung des Systems, basierend auf einer service-orientierten Architektur und die technische Umsetzung durch Web Services. Des Weiteren sind Analyse, Simulation und Monitoring der Prozesse Aufgaben, die für ein solches System notwendig sind.

Aufgrund der zunehmenden Globalisierung in der Hochschulausbildung – einerseits mit Hilfe des weltweiten Informationsnetzwerkes und andererseits aufgrund des internationalen Wettbewerbs zwischen Universitäten – muss die IT-Unterstützung für junge Wissenschaftler initiiert werden. Das aus dem vorgestellten Ansatz resultierende System wird eine Basis bilden, um eine umfassende Informations- und Kommunikationsinfrastruktur mit hohem Informations-, Planungs- und Management-Potenzial aufzubauen.

Literatur

[EU99] The Bologna Declaration. Joint declaration of the European Ministers of Education. Bologna, 19. Juni 1999.

[EU03] Realising the European Higher Education Area. Communiqué of the Conference of Ministers responsible for Higher Education. Berlin, 19. September 2003.

[EU06] Programmes and Actions of the European Commission. http://ec.europa.eu/education/programmes/programmes_en.html [letzter Abruf am 08.09.2006].

[FW04] Fallside, D.C.; Walmsley, P. (Hrsg.): XML Schema Part 0: Primer Second Edition. W3C Recommendation. World Wide Web Consortium, 28. Oktober 2004. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/primer.html [letzter Abruf am 08.09.2006].

[GL81] Genrich, H.J.; Lautenbach, K.: "System modelling with high level petri nets". In Theoretical Computer Science, (13):109-136, 1981.

[Je93] Jensen, K.: "An Introduction to the Theoretical Aspects of Coloured Petri Nets". In de Bakker, J.; de Roever, W.P.; Rozenberg, G. (Hrsg.): A Decade of Concurrency – Reflections and Perspectives. Lecture Notes in Computer Science, Nr. 803. Springer, 1993; S. 230-272.

[Ju06] Juling, W. et al.: "Architektur für ein universitätsweit integriertes Informations- und Dienstmanagement". In Proceedings des Workshops „Pervasive University" im Rahmen der 36. GI-Jahrestagung "Informatik 2006", 02.-06. Oktober 2006 [zur Veröffentlichung angenommen].

[Ke05] Kehm, B.: "Developing Doctoral Degrees and Qualifications in Europe. Good Practice and Issues of Concern". In Beiträge zur Hochschulforschung, 27(1):10-33, 2005.

[LO03] Lenz, K.; Oberweis, A.: "Interorganizational Business Process Management with XML Nets". In Ehrig, H. et al. (Hrsg.): Petri Net Technology for Communication Based Systems. Lecture Notes in Computer Science, Nr. 2472. Springer, 2003; S. 243-266.

[LO04] Lenz, K.; Oberweis, A.: "Workflow Services: A Petri Net-Based Approach to Web Services". In Proceedings of the International Symposium on Leveraging Applications of Formal Methods (ISoLA 2004), Paphos, Cyprus, November 2004. Technical Report, Nr. TR-2004-6. Department of Computer Science, University of Cyprus, 2004; S. 35-42.

[SN00] Scheer, A.-W.; Nüttgens, M.: "ARIS Architecture and Reference Models for Business Process Management". In van der Aalst, W.M.P.; Desel, J.; Oberweis, A. (Hrsg.): Business Process Management – Models, Techniques and Empirical Studies. Lecture Notes in Computer Science, Nr. 1806. Springer, 2000; S. 376-389.

[Re86] Reisig, W.: Petrinetze – Eine Einführung. Studienreihe Informatik. Springer, 2. Auflage, 1986.

[RR98] Reisig, W.; Rozenberg, G. (Hrsg.): Lectures on Petri Nets I: Basic Models. Lecture Notes in Computer Science, Nr. 1491, Springer, 1998.

[VT06] Vossen, G.; Thies, G.: "ISETTA: Service Orientation in the "Bologna Process" of a Large University". In Nilsson, A.G. et al. (Hrsg.): Advances in Information Systems Development: Bridging the Gap between Academia & Industry. Springer, 2006.

[VW03] Vossen, G.; Westerkamp, P.: "E-Learning as a Web Service (Extended Abstract)". In Desai, B.C.; Ng, W. (Hrsg.): Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS 2003). IEEE, 2003; S. 242-249.

Implementing a Service-Oriented Architecture for Small and Medium Organisations

Pascal Bauler, Fernand Feltz, Nicolas Biri, Philippe Pinheiro

Department Informatics, Systems, Collaboration (ISC)
Centre de Recherche Public – Gabriel Lippmann
rue du Brill 41
4422 Belvaux, Luxembourg
{bauler, feltz, biri, pinheiro}@lippmann.lu

Abstract: This paper explains how we have designed a service-oriented architecture by combining and extending several open source environments. We show how the IT architecture of the National Family Benefits Fund in Luxembourg got systematically modernized by solving issues related to Enterprise Application Integration, before introducing concepts like Enterprise Service Bus which was extended to offer service orchestration and business workflow support. By taking advantage of Model Driven Software Design and by introducing a framework for adding new components to the service bus, technical burdens to get used to the new IT environment got minimized. The cost saving due to the open source sub-systems and the ease of use make the proposed architecture particularly interesting for small and medium organizations.

1 Introduction

Integration issues are topics that are mainly addressed in large companies or administrations. However the problems related to heterogeneous IT environments are also well known in Small and Medium Enterprises (SMEs) or Small and Medium Administrations (SMAs). In this paper we explain various research results we collected from projects with actors in Luxembourg and in particular the Luxembourg National Family Benefits Fund (Caisse Nationale des Prestations Familiales) (CNPF) and we propose an approach how to tackle integration problems in SMEs and SMAs with limited budgets.

Due to the decentralisation of the IT environments and the small size of the country, even central administration services in Luxembourg have to be considered as SMAs, with a headcount of about 100 full time equivalents.

In the following sections, we explain how the IT environment of the CNPF got modernized in a step-by-step approach. First of all, technical issues related to Enterprise Application Integration (EAI) got solved by integrating various heterogeneous and distributed IT solutions ([HW04] and [Li03]). In a second phase we designed an Enterprise Service Bus (ESB) offering standards based connectivity, integration, routing and transformation of data. ESBs may consist of distributed but interconnected sub-components, which tackle by design issues like high availability and reliability (see reference [Ch04] pages 7-10). The proposed ESB assures the technical integration of existing applications covering Mainframe applications, Document Management Systems, custom developed applications by means of various technologies and classical relational databases. In a third phase, a business-oriented integration was realized by adding business workflow support and by setting up a complete Service-Oriented Architecture (SOA). When moving towards a SOA, the enterprise wide IT architecture is reoriented towards a service-oriented approach. As specified in [Oe05], a SOA has a significant business impact by granting access to complete business functions and workflows through services, which may be distributed over the entire company or administration.

As the proposed solution is focused onto SMEs and SMAs, the ease of integration and usage of distributed services is crucial. In order to achieve this goal we use Model Driven Software Development techniques (MDSD) as defined in [SV05] and we propose a generic development framework to develop new ESB components. Dedicated meta-models and development artefacts hide all technical integration issues encountered when accessing the SOA or when making new services available.

The proposed solution is a starting point for new horizontal applications, which take advantage of existing distributed and heterogeneous services in order to offer new business functions.

2 Moving towards a Service-Oriented Architecture (SOA)

Based on the explanations of the previous section, we took a 3-step approach. First of all we started with a technical integration of existing applications. These integration modules were reused in a second phase as connectors which linkup existing applications to the service bus. Once these components stabilized, we integrated a workflow management system able to run business workflows. In the coming paragraphs we will discuss the various topics in detail.

2.1 Application integration and connectors

During the first project phase, we focused onto the question on how to retrieve and to inject data into existing applications running at our partner's site. A first inventory of existing applications resulted in a list of heterogeneous applications, using different technologies, design paradigms and offering different integration possibilities, which could be grouped into 4 categories:

- Mainframe applications based on ISAM[1] files
- A document management system with clearly documented APIs
- Classical Client/Server applications accessible through JDBC[2]
- Custom built applications accessible through DLLs, Web Services or monolithical WinDev[3] applications. Integration of WinDev applications is particularly tricky as this development environment is typically used in departmental environments where integration issues are of little importance.

The design goal in this first project phase was to work out technical solutions able to get read/write access to all relevant data. This goal could only be achieved by combining several technical approaches and by doing some reverse engineering on poorly documented components. At the end of this phase, we had reliable and standardized interfaces to all existing business applications running in production. To minimise development efforts, existing interfaces available as DLLs, or WinDev modules, were reused and encapsulated by means of C++ and Java technologies, in order to guarantee transaction safety. The total number of interactions and interfaces between the various sub-systems is (n*(n-1)/2), where n is the number of implied systems. In the concrete scenario of the Luxembourg National Family Benefits Fund, described in this document, the number of implied systems is 9 as shown on figure 1 (2 mainframe applications, 1 document management system, 1 JDBC based client/server application and 5 custom built systems), which results in up to 36 interfaces between the various sub-systems. To fight this complexity, the concept of ESB was introduced and offered a standardized way of integrating the various connectors.

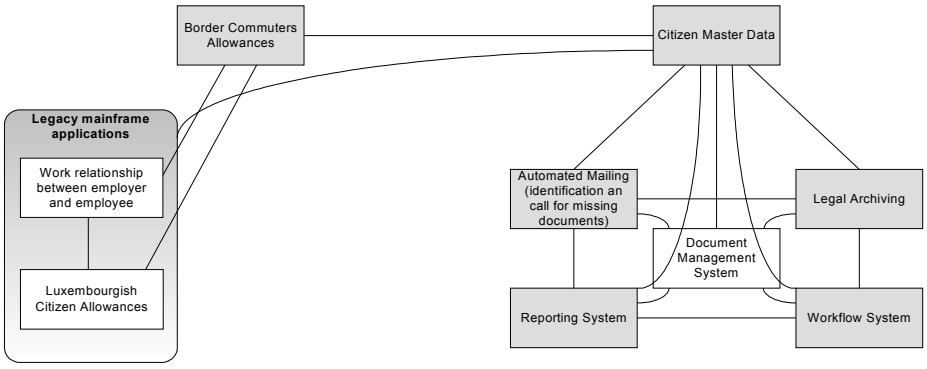


Figure 1. Initial system landscape at the CNPF

[1] ISAM: Indexed Sequential Access Method
[2] JDBC: Java Database Connectivity as defined by SUN Microsystems (http://java.sun.com/javase/technologies/database.jsp)
[3] WinDev 10 is a commercial rapid application development (RAD) environment conceived by PC-Soft (http://www.pcsoft.fr/windev/index.html)

2.2 Enterprise Service Bus (ESB)

The main benefit of this concept is its ability to handle all topics related to message routing, communication protocols and application integration in a central standardized way. Rather than establishing direct communication between EAI interfaces, or Web Services and a Workflow Management System as mentioned in [MKM05], we decided to introduce a Java Business Integration (JBI) [TW05] based ESB.

Indeed, the JBI specification and API define a platform for building enterprise-class ESBs using a pluggable, service-based design. The JBI runtime core mainly comprises the following components within the same Java Virtual Machine:

- Component framework enabling the deployment of the different types of components within the JBI runtime.
- Normalized Message Router [Vi05] enabling a standard mechanism of message interchange between the services. The messaging model uses the message exchange patterns defined in the Web Services Description Language (WSDL) [W306].
- Management framework based on Java Management Extensions (JMX) [KHW02] enabling the deployment, management and monitoring of components within the JBI runtime.

The JBI environment hosts plug-in components such as transport binding, routing engines, rule engines and transformation services. JBI defines two types of components: Service Engine Components (SE) responsible for implementing business logic and other services, Binding Components (BC) used to provide transport level bindings for the deployed services.

In order to deliver a service, each component needs a deployment package, called Service Unit (SU), containing deployment information. This information is composed of JBI standard information indicating consumed and produced services, as well as component specific information, proprietary for that particular component. Some complex components may require extra artefacts in order to provide a certain service (the Business Process Execution Language (BPEL) [An05] component for instance, relies on BPEL processes, which correspond to this type of artefacts). Such artefacts are packaged into the Service Unit.

In order to deploy SUs in a JBI container, they have to be grouped into packages offering composite services. These packages are called Service Assemblies (SA).

After thoroughly comparing the available solutions, we noticed that several stable open source solutions were available and, rather than developing a JBI container from scratch or buying a commercial solution, we decided to focus onto ServiceMix[4]. To guarantee platform independency only JBI standard functionalities were used and proprietary ServiceMix extensions were avoided. A consequence was the free choice of the JBI container to be used in the partner's IT environment.

The next step was to route all remote data accesses throughout this central system and, as a consequence, reduce the number of required interfaces. In this new design all components access the service bus through appropriate connectors or JBI Binding Components, when using the JBI terminology. A next step was to guarantee transport layer independency when accessing the various binding components. Dedicated transformation components were introduced, which act as access points for JBI client applications. These components accept connections using JMS[5], SOAP[6] or the Http Request protocol. The transformation components decompose incoming data and forward them to the appropriate components to access external applications. Any new client application has free choice to access all available services through any of the 3 above-mentioned protocols. In the current design we considered 2 types of binding components:

- Access points for external client applications that handle user requests, using any of the supported formats. Incoming requests are automatically transformed into appropriate JBI messages and forwarded to 'server side' binding components.

- Server side components accept JBI messages from the container and establish reliable data exchanges with external server applications.

[4] ServiceMix – an Open Source based *"Enterprise Service Bus"*, http://www.servicemix.org
[5] JMS: Java Message Service, a Java based framework for handling data exchanges through message queues
[6] SOAP: Simple Object Access Protocol, the underlying transport protocol of Web Services

All binding components required by our external partner (CNPF), have been realized and integrated into the above-mentioned environment. To guarantee platform independency, special care was taken to develop fully JBI compliant components without using any platform specific APIs. By doing so, we are free to deploy our solution and architecture in any JBI compliant IT landscape (as validated by replacing for test purposes ServiceMix by OpenESB[7]). The above-mentioned architecture offers a middleware acting as Enterprise Service Bus. By introducing the concept of transport protocol independency and a clear separation between client and server components, several technical issues became transparent and the complexity of using the proposed ESB got reduced. As all efforts have been based on Open Source technologies, initial costs of the ESB are minimal and the proposed solution is particularly interesting for SMEs and SMAs. Custom developed components have to be covered by a standard maintenance contract. Issues like high availability and geographical spreading of the service bus, which are handled by the JBI specification, were not relevant by now, as we concentrated onto SMEs and SMAs where similar constraints were not encountered. A next project phase was to extend this architecture into a SOA by making business workflows available throughout services.

2.3 Service-Oriented Architecture (SOA)

A prerequisite of running efficient business workflows is to add service orchestration to the JBI container. Functionality offered by several independent and distributed services are combined in business workflows orchestrated by the Workflow Management System. These business processes are made accessible as newly available services, available to the whole IT environment. When conceiving an orchestration service 4 key aspects were relevant for our project:

- Standards based Workflow management system.
- Existence of efficient and user-friendly modelling tools.
- Possibility to execute modelled workflows. As the workflows are supposed to be used in an existing operational IT environment, easy execution of the workflows is mandatory.
- Smooth integration into the JBI container.

[7] OpenESB– an Open Source based *"Enterprise Service Bus"*, https://open-esb.dev.java.net

After analysing the available standards and environments, we decided to aim for Business Process Execution Language (BPEL), which seems to be a very promising approach, with several implementations available. BPEL is specially suited for integration with JBI containers as both technologies rely on WSDL as service description language. After comparing several BPEL engines, ActiveBPEL[8] was retained and integrated. ActiveBPEL is an Open Source BPEL engine, which comes with user-friendly modelling tools. Compared to other free BPEL engines ActiveBPEL seems to be the most stabilized package. Full integration of ActiveBPEL as orchestration service into the JBI architecture required the usage of native ActiveBPEL APIs during the development process.

By definition, all communications with a BPEL engine are handled throughout web services using the SOAP protocol. By doing so, the BPEL engine is completely independent of the JBI container. This approach introduces a significant overhead, which is due to message transformations when exchanging data between the JBI container and the BPEL engine and vice-versa, and to network latency when initiating communication between JBI container and BPEL engine. To avoid these performance issues, a native integration of ActiveBPEL into ServiceMix was realized. As a consequence BPEL is able to access other JBI components directly by means of JBI messages. JBI components in charge of orchestrating communication between various other components are Service Engines (when using the JBI terminology). As there might be some controversy about allowing access to BPEL workflow throughout JBI by avoiding the SOAP transport protocol, we decided to make both options available. After integrating a BPEL engine, all technical pre-requisites for a SOA are fulfilled (figure 2 gives a detailed overview of the finalized SOA). To facilitate the administration of the SOA container, a JMX based monitoring tool was added. A Spring [Jo05] based graphical user interface was designed, to display JMX interfaces related to the JBI container and to the JBI components like ActiveBPEL and other custom developed components.

[8] ActiveBPEL – *"The Open Source BPEL Engine"*, http://www.activebpel.org

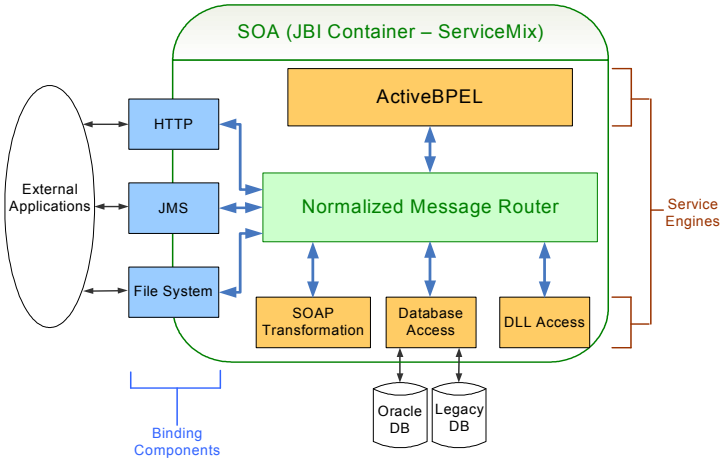


Figure 2. General overview of the proposed SOA

In the following sections we will briefly describe our efforts to facilitate access to the various services and the JBI container.

3 JBI component framework

According to the JBI specification, each component must implement the standard JBI API. After the development of a few JBI components, we noticed that many code segments were duplicated. In order to facilitate the development of new JBI components and to let the component developers concentrate onto the business logic, we proposed a common framework. This common framework eliminates code duplication and offers a coherent architecture for future JBI components. Furthermore, maintenance and improvement of the components are simplified. In this section we describe the proposed component framework and its practical usage by illustrating the design of two newly developed JBI Components. The selected components are the JMS transport binding component, taking care of routing/transporting JMS messages, and the "Report Factory" service engine, a generic reporting engine used to manage and execute queries on relational databases or document management systems. When explaining the various design choices, we frequently reference design patterns specified by the Gang of Four [Ga97].

The central element of the proposed framework, called the "component core", corresponds to the generic core features common to all JBI components. It is also in charge of coordinating the other elements of the framework. It takes advantage of the template method design pattern and it provides an abstract implementation of the JBI APIs mandatory for every JBI component. This component core guarantees that all components share the same basic behaviour towards the JBI container and component specificities can be added by extending this initial template.

Another key element of the framework is the “message handler”. This element handles incoming and outgoing messages transiting between the components and the JBI container. It provides a basic implementation of the message exchange patterns [W306]. This implementation verifies the validity of incoming messages and keeps track of the active exchanges involving the component. The message handler is developed according to the facade design pattern such that components can easily bind message reception to business functionalities. We may cite for instance the Report Factory component, where we can either add new queries or execute existing ones. These two operations correspond to business functions registered in the message handler. When a new message is received, the message handler checks its content to determine if a query must be stored or executed and call the appropriate function.

According to the JBI specification, each component requires a standard XML configuration file in order to be deployed in a JBI container and each valid Service Unit (defined in section 2.2) must contain an XML deployment descriptor. As the components are in charge of parsing those descriptors, a “configuration” element was added to the framework. The main role of this element is to parse the XML documents and convert them into valid Java objects. The JBI specification allows the extension of the XML configuration file. Such extensions can be used to tune specific aspects of the component. The “configuration” element parses the JBI standard configuration file and takes advantage of the decorator design pattern, which facilitates the parsing of component specific properties. For instance, in the JMS transport component, we need to define the receiving and sending queues of JMS messages. These properties are set in the component specific parts of the XML configuration files.

The JBI specification also defines a component management interface by means of JMX. There are two possible management levels, a standard one defined in JBI API (required) and a component specific one (optional). Consequently, a common standard JMX management interface is defined in the framework. This interface can be extended with component specific management functionalities. An example of component specific JMX management can be found in the Report Factory service engine. For this component we enable the dynamic parameterisation of the database connection, by specifying whether connections are pooled and by setting an optional pool size.

Finally, every JBI component must be able to deliver the metadata describing each service it provides, which corresponds to the WSDL service description. A common generator was defined at framework level. Each JBI component is free to add additional information to the generated WSDL by exploring the content of its JBI configuration files.

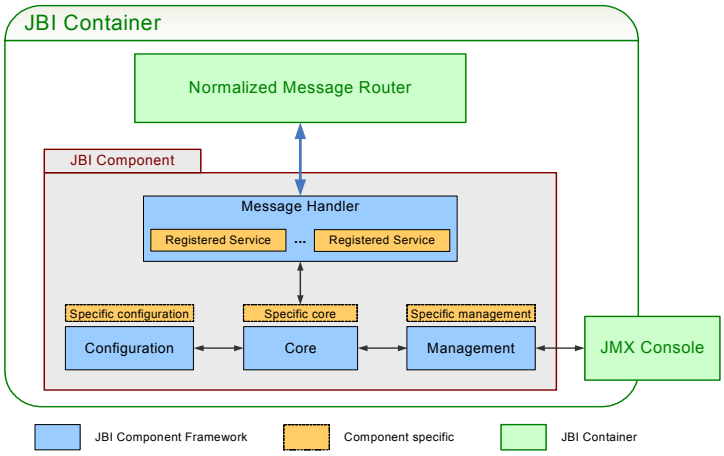


Figure 3. Architecture of a component built with the JBI Component framework

All these elements put together (as shown on figure 3) considerably reduce the development effort and technical complexity when conceiving new JBI compliant components. Furthermore, the proposed framework handles most of the JBI specific requirements and the development efforts are concentrated onto business specific requirements. Through several practical examples, we experimented the efficiency of the design patterns used in the proposed JBI component framework. It considerably facilitated the development of new JBI components and the integration of the proposed architecture in a large variety of integration scenarios.

4 SOA and Model Driven Software Development

When developing business applications taking advantage of a SOA, the source code can be split into two parts:

- *Business code*, realizing features like invoices, accounting information
- *Technical code*, handling problems like data persistency or access to the SOA container

Business code is usually domain-specific whereas technical code is more generic, containing a lot of repetitions and also requiring a detailed knowledge of underlying technologies. In order to overcome this problem we used a Model Driven Software Development (MDSD) approach to generate the technical code out of UML models. In practice the business application is modelled by means of UML diagrams. These diagrams are adjusted in order to fulfill the constraints imposed by the meta-model. When running the generators, the technical code is completely generated out of these UML models, as well as the skeleton of the business logic. Business developers complete the business logic manually by following appropriate design patterns like abstract factories and decorators [Ga97], in order to avoid problems related to protected regions in the generated code segments. The current meta model is specially fitted for EJB 3.0[9] and offers full integration with the JBI container. By adding specific tagged values and stereotypes into the UML models, you can automatically access remote and distributed services throughout the JBI container, or make business functions available as new web services. The code generator will generate the appropriate EJB code segments and annotations, JBI specific client- or server-oriented binding components as well as appropriate deployment information. As a consequence, SOA services can be transparently accessed and created by business logic developers and all technical complexity is hidden. OpenArchitectureWare[10], an open source framework, is used to offer MDSD functionality to the business logic development teams.

5 Use case at the Luxembourg National Family Benefits Fund

A first major horizontal application taking advantage of the SOA handles the payment of the family benefits in Luxembourg. Integration has to be established with the following sub-systems: a WinDev application managing master data, the mainframe holding administrative data, as well as a communication sub-system in charge of handling physical data exchanges with the French Family Allowances Agencies (CAF) and the banks to execute the payment orders. This application is currently in a testing phase and will be fully exploited in production beginning 2007 to handle the French border commuters.

[9] EJB: Enterprise Java Beans as specified in the JSR 220 http://java.sun.com/products/ejb/docs.html
[10] OpenArchitectureWare - *"the flexible open-source tool platform for model-driven software design"*, http://www.openarchitectureware.org

Major challenges when moving to a SOA are related to organisational issues and to a conceptual change in the perception of how to organize and optimize business processes. The Luxembourg National Family Benefits Fund, like many SMEs and SMAs, works on a paper-based approach, without having clearly defined business processes. Even projects like the installation of a Document Management System did not modify this initial concept, except that paper based files were replaced by scanned documents. As a consequence, a first task was to identify underlying business processes and to implement those processes in the SOA. During this Business Process Reengineering (BPR) phase, we used ARIS[11] EPC[12] as a communication tool between business and IT people. EPC was then used as basis for creating BPEL workflows, which on their side are too technical for business people. The business applications were modelled by means of UML respecting the constraints imposed by the meta-model around those BPEL processes. The code generator created the technical code segments (EJB, SOA and JBI related) hiding the technical complexity to the application developers. The new system landscape is detailed in figure 4.

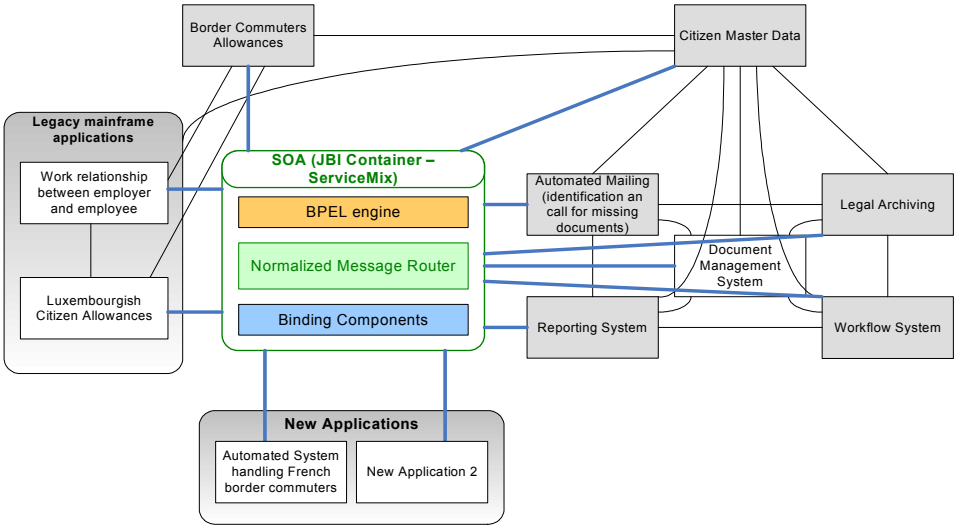


Figure 4. New system landscape at the CNPF

[11] ARIS: an modelling toolset provided by IDS-Scheer http://www.ids-scheer.com

[12] EPC: Event Driven Process Chain

6 Conclusion

In this document we give a rough overview on how we modernized the IT environment of the Luxembourg National Family Benefits Fund by introducing the concept of Service-Oriented Architecture. The proposed approach combines and extends several standards based open source solutions to systematically move from a EAI approach to the concept of Enterprise Service Bus respecting the JBI standard, to finally integrate service orchestration and business workflow support through a BPEL engine. We also show how this new IT environment gave life to new types of horizontal applications and how MDSD techniques facilitated the usage of this new environment.

Our contribution consists in realising the complete integration of the BPEL workflow engine into the JBI container as well as the design of a JBI development framework and a meta-model by means of MDSD techniques, in order to facilitate development of new and usage of existing JBI components. In addition, the management framework for the JBI container and BPEL engine were integrated and extended. This new environment was adapted to our partner's IT environment by developing adequate connectors and interfaces to existing business applications and by establishing a new development process, specially adapted to the Luxembourg National Family Benefits Fund.

The next step is to analyse how the SOA is deployed and help our external partner to take maximum advantage of the new possibilities. In parallel we will study other emerging integration initiatives like Service Component Architecture (SCA) [Be05], standards, which offer additional levels of platform and development language independency. We also plan to analyse how a higher level SOA architecture, as defined in [BGR05], can be integrated into our platform.

Acknowledgements

The authors want to thank Michel Neyens and Claude Nicolas of the 'Caisse Nationale des Prestations Familiales' for their support and collaboration in this project.

References

[An05] Andrews T., Curbera F., Dholakia H., Goland Y., Klein J., Leymann F., Liu K., Roller D., Smith D., Thatte S., Trickovic I., Weerawarana S., Business Process Execution Language for Web Services (BPEL4WS), ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, May 2005.

[Be05] Beisiegel M. Blohm H., Booz D., Dubray J.-J., Edwards M., Flood B., Ge B., Hurley O., Kearns D., Lehmann M., Marino J. Nally M. Pavlik G., Rowley M., Sakala A., Sharp C., Tam K., Service Component Architecture – Assembly Model Specification (version 0.9), http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-sca/SCA_AssemblyModel_V09.pdf, November 2005.

[BGR05] Berbner R., Grollius T., Repp N., An approach for the Management of Service-oriented Architecture (SoA) based Application Systems, in Enterprise Modelling and Information Systems Architectures 2005 (EMISA 2005), p. 208-221, October 2005.

[Ch04] Chappel, D.A.: Enterprise Service Bus. O'Reilly Media, 2004.

[Ga97] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, 1997.

[HW04] Hohpe, G., Woolf, B., Enterprise Integration Patterns, Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, 2004.

[Jo05] Johnson R., Hoeller J., Arendsen A., Risberg T., Sampaleanu C., Professional Java Development with the Spring Framework, Wiley, 2005.

[KHW02] Kreger, H., Harold, W., Williamson, L., Java and JMX, Building Manageable Systems, Addison-Wesley, 2002.

[Li03] Linthicum, D.S., Enterprise Application Integration, Addison-Wesley, 2003.

[MKM05] Müller, J., Kümpel, A., Müller, P., Integration und Orchestrierung von Geschäftsprozessen in Web Applikationen – Ein Service-Orientierter Ansatz, in Entreprise Application Integration 2005 (EAI 2005), http://www.ceur-ws.org/Vol-141/paper10.pdf, july 2005.

[Oe05] Oey, K.J., Wagner, H., Rehbach, S., Bachmann, A., Mehr als alter Wein in neuen Schläuchen: Eine einführende Darstellung des Konzepts der serviceorientierten Architekturen, Enterprise Architekture. Unternehmensarchitekturen und Systemintegration, Band 3, Gito-Verlag, 2005.

[SV05] Stahl, T., Völter, M., Modellgetriebene Softwareentwicklung, Techniken, Engineering, Management, dpunkt.verlag, 2005.

[TW05] Ten-Hove, R., Walker P., Sun Microsystems, JSR 208: Java Business Integration (JBI), http://www.jcp.org/en/jsr/detail?id=208, final release, august 2005.

[Vi05] Vinoski, S., Towards Integration – Java Business Integration, IEEE Internet computing, http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?pName=dso_level1&path=dsonline%2F0507&file=w4tow.xml&xsl=article.xsl, July 2005.

[W306] W3C, Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts, http://www.w3.org/TR/wsdl20-adjuncts, candidate recommendation, March 2006.

Challenges and Solutions in Planning Information Systems for Networked Value Constellations

Novica Zarvić* Maya Daneva†

University of Twente, Department of Computer Science, Information Systems Group
P.O. Box 217, 7500 AE Enschede, The Netherlands
{n.zarvic, m.daneva}@ewi.utwente.nl

Abstract: Nowadays businesses often decide to form networked value constellations in order to satisfy complex customer needs. To fulfill the value-based requirements of an e-Business idea and to realize the coordination of such a multi-actor network an adequate underlying information systems architecture has to be conceptualized. This paper discusses the applicability of classical information system planning approaches, such as Information Engineering to cross-organizational settings expressed through value-based requirements. On the basis of this analysis several requirements for the enhancement and adaptation of Information Engineering-like methodologies for e-Business ideas are defined for the purpose of enabling alignment between a value-based business context and the information systems architecture in a networked environment. The paper proposes a way to derive data-orientation from value-orientation, i.e. an enterprise model from a value model. This in turn enables afterwards the straightforward use of traditional data-oriented techniques for value-based business models.

1 Introduction

The biggest challenge in information systems (IS) planning and design is the alignment of IS to the business context [WBFG03]. The adequate allocation of IS to business needs must be considered as one of the fundamental factors in IS development, because only a well coordinated IS landscape will lead to the fulfillment of the given business goals. Coordination of IS plans and business plans is not a new problem and the difficulty of achieving it was addressed in most organizations by using structured IS planning methodologies [LM89]. In businesses of any significant size, however this problem is not solved completely. With the advent of the internet many profit-and-loss responsible business units, or often independent companies, form so-called *networked value constellations*, which makes the situation even more difficult [WGvE05], because most businesses cannot be viewed anymore as single enterprises. Such networked cooperations are enabled

*Supported by the Netherlands Organisation for Scientific Research (NWO), project 638.003.407, Value-based Business-IT Alignment (VITAL)

†Supported by the Netherlands Organisation for Scientific Research (NWO), project 632.000.000.05N01, Collaborative Alignment of cRoss-organizational ERP Systems (CARES)

by means of ICT (information and communication technology) and it is the goal of this paper to explore how to match ICT-services with the requirements of a value-based business model. Typically in networked value constellations almost all services are e-services, services produced by ICT-systems. Examples are on-line music delivery, e-mail services, and information services.

Tagg and Freyberg argue that "no methodology, however well-championed, has ever kept a monopoly on the truth for very long. Both the Technology and Business Application ends of IS have been so fast moving that the 'design equation' is always changing" [TF97]. This makes it clear that the IS 'design equation' has an evolutionary character, i.e. existing methodologies are permanently (depending on predefined or evolving requirements) adjusted and enhanced [Bin96].

This paper is structured as follows. Section 2 gives an introduction to ISP methodologies, especially to Information Engineering. The notion of networked value constellations is described in detail and the adequacy of Information Engineering for such constellations is discussed. Section 3 defines the additional steps needed to enhance existing methodologies, before in Sec. 4 the main artifacts of an enterprise model are derived from a value model. Finally, Sec. 5 concludes the paper.

2 Information Systems Planning Methodologies and Networked Value Constellations

Strategic planning for information systems is the discipline to position an organization to proactively take advantage of its future by anticipating opportunities and challenges in a dynamic and complex environment and by providing the roadmaps that master change. In today's interconnected business world, strategic planning is the vehicle organizations use to attain business/IT alignment. Recent empirical studies [CST06, BH06, RB00] indicate that planning directly impacts the development of shared business-IT knowledge, and, consequently, alignment between business and IT strategy. For network businesses, the purpose of strategic planning processes is to bring business and IS senior managers from partner companies together repeatedly in identifying, assessing, and realizing technology-related opportunities and managing challenges. Although this is well recognized as a key to the successful implementation of inter-organizational information systems, very little empirical and theoretical research has been done on strategic planning processes for inter-organizational systems [FGP03]. As a result, practitioners are left with little coherent planning advice for approaching network constellations. In the next sections, we seek to address this deficit. We provide a summary on the family of Information Engineering-like methodologies, analyze why existing approaches offer little support to systems planners in business networks, and define what should be included in new, enhanced approaches to inter-organizational systems planning.

2.1 Summary of Information Engineering

Many different IS planning methodologies are discussed in the literature [PS95, PR01] as well as the need for designing new methodologies or adapting existing approaches to specific environments and requirements [IA05, WG98]. The ability to involve top management in the systems planning activity is an important strength [Mar82] and has the aim to ensure that the information systems direction is linked to the business direction. In this paper we focus on an IS planning methodology called *Information Engineering* (IE). IE was initially created by Clive Finkelstein [Fin89] who is known as the "father of information engineering". However, other associate the term IE with James Martin who popularized this methodology during the 1980's through various publications [Mar89] and through his consulting company "James Martin Associates". It became then a *de facto* standard in IS planning and this explains that also major computer companies like Texas Instruments [Bin96] acted successfully in this field. James Martin defined Information Engineering as "The application of an interlocking set of formal techniques for the planning, analysis, design, and construction of information systems on an enterprise-wide basis or across a major sector of the enterprise [Mar89]."

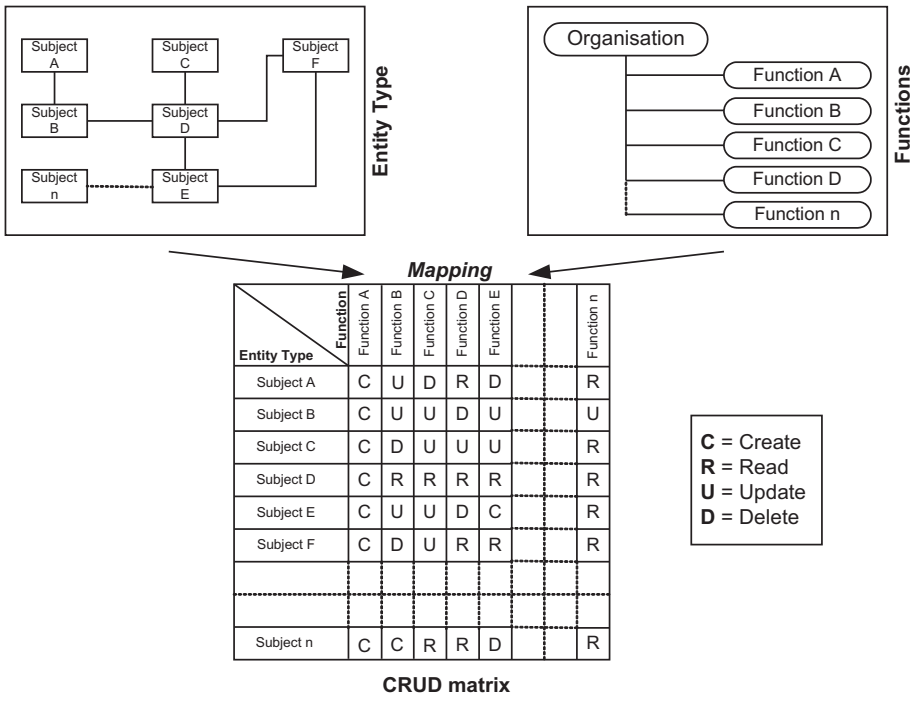


Figure 1: Mapping entity types against functions by means of a CRUD matrix

This definition makes the different steps of IE clear, which are often represented with

the help of the information systems pyramid. In short, IE is concerned with the creation of *enterprise models*, *data models* and *process models* in a top-down manner. The last step usually requires that functions/processes need to be divided and allocated to systems that support them. In case of small companies with few functions this can be realized by looking which IS are supporting the identified functions. But usually there are hundreds of functions and in such case the systems are determined by means of CRUD analysis, which takes subject areas and business functions into account. Subject areas are entities about which an enterprise wants to save information (e.g. products, customers). A business function is a group of activities in an enterprise (e.g. purchasing, receiving). Business functions are sometimes grouped into higher-level functional areas [Mar89]. Tagg and Freyberg call *clustering* the "engine room" of most IS planning methods which aids the identification of smaller units which are relatively self-contained. In order to do so the entities need to be mapped against functions like shown in Fig. 1. The letters C, R, U or D in the cells of the CRUD matrix show whether a function *creates*, *reads*, *updates* or *deletes* instances of an entity. Clustering itself is a process of reordering the rows and columns of the CRUD matrix such that the most C, D and U cells are grouped into rectangular blocks like shown exemplary in Fig. 2. These rectangular blocks represent natural systems and point out the relationship between data subjects and functions. Note that IS planning usually ends up with the identification of such systems [TF97].

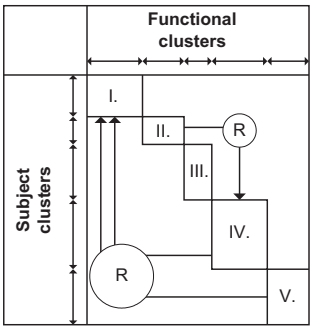


Figure 2: Clustered CRUD matrix (cf. [TF97])

Although IE as a methodological approach is overaged today [Hay03], many techniques from the traditional IE methodology are still used in practice. The term itself disappeared in the middle of the 1990's from the scene, but was revived through various "new" Information Engineering study courses at different academic institutions (e.g. Hogeschool van Amsterdam in the Netherlands [1], National Taiwan University [2], Australian National University [3], University of Osnabrück in Germany [4]) throughout the world. Note that the curricula of this study courses do not have necessarily much in common with the primarily IE like presented by James Martin.

[1] See http://www.iie.hva.nl/
[2] See http://www.csie.ntu.edu.tw/
[3] See http://rsise.anu.edu.au/
[4] See http://www.inf-eng.org/

2.2 The Context of Networked Value Constellations

Networked value constellations are networked businesses that focus especially on how economic value is created and exchanged within a network of actors. For the purpose of this paper we borrow Santana Tapia's definition of a networked business: it is "a web of profit-and-loss oriented business units, or of independent companies, that work together for a common purpose on a specific period of time" [Tap06]. Such a network is enabled by means of ICT [WGvE05]. An example of a business network is WalMart Stores Inc. who collaborates with a large number of non-U.S. companies and gives them direct access to the American market. In the context of a networked business, system planning begins as a learning process where partner companies are studying inter-organizational support technologies and how they can benefit from these technologies in specific circumstances [FGP03]. This helps conceptualize a value exchange model that serves as a business argument used later to devise cross-organizational coordination processes and systems to support them. The discipline of requirements engineering for such systems postulates that inter-organizational IS analysis and design includes three distinct viewpoints on the system under consideration [GA01, GA03], a *value viewpoint*, a *process viewpoint*, and a *IS viewpoint*. The value viewpoint is the top-level viewpoint and focuses on value creation, distribution and consumption. It is usually the first step in designing and analyzing a networked value constellation. Gordijn proposes the e^3 - *value* methodology as a representation technique for modelling the business from a value viewpoint. The other viewpoints to be considered are the process viewpoint and the IS viewpoint. All the viewpoints represent different *Universes of Discourse*, but are nevertheless describing the same sociotechnical system, and therefore mutually adapt to each other. This means that a change in one viewpoint will most probably lead to changes in the other two viewpoints. The main relationship between these viewpoints is a *put into operation relation*, that is, the system presented through the IS viewpoint puts into operation the processes described in the process viewpoint which - in turn, put into operation the value-based business model. Following these authors [GA01, GA03] our emphasis in this paper is on the *value viewpoint*.

Figure 3 shows an example networked value constellation by Derzsi and Gordijn [DG06, p. 222] using the e^3 - *value* notation. The example describes the so-called "travelling salesman" - case. In this case a salesman travels internationally and wants to have wireless Internet access in the hotels, where he is residing. As far as the e^3 - *value* notation is suited for describing the flow of *goods* and/or *services*, the wireless internet connection represents in this case a service and the room which the salesman rents represents a good. Of course the hotel does not provide the salesman these value objects for free, but demands something of value in return (fee). In this case the hotel offers the wireless connectivity by itself and uses an internet service provider (ISP) to have internet connectivity.

In short, in e^3 - *value* a business *actor* is a profit-and-loss responsible business unit, like the hotel in our example case. It is represented by means of a rectangle. A *market segment* is a group of actors that share the same needs and is represented as three stacked actors such as the traveller. Actors and market segments exchange *value objects* (e.g. money for IP access). This is realized through *value ports*, which are located at the *value interfaces*. A value interface consists of *in* and *out* ports that belong to the same actor or market

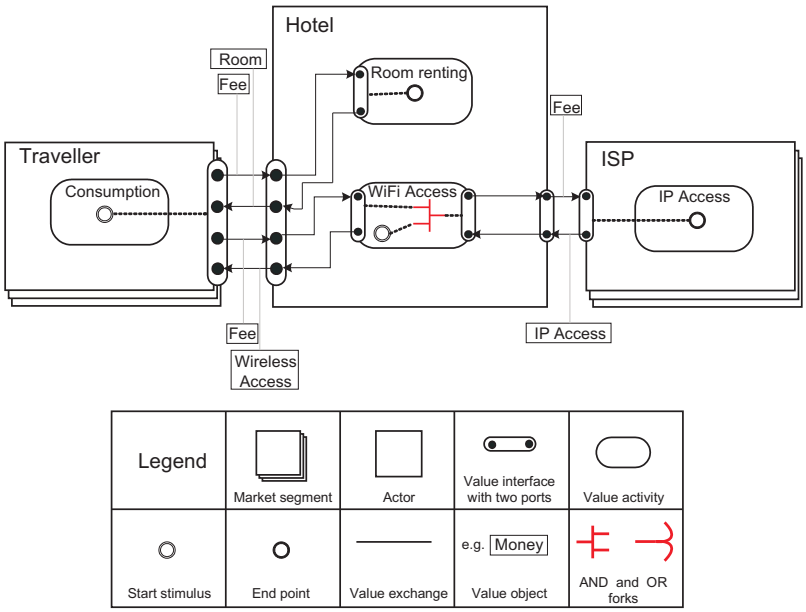


Figure 3: Example e^3 - *value* model "Travelling Salesman"

segment. A *value exchange* is used to connect two value ports with each other and is shown as line between the value ports. A *value activity* is graphically represented as a rounded box drawn inside an actor or market segment, like e.g. WiFi access in Hotel. It represents assignment of value activities to actors. A scenario path begins with a *start stimulus*, which in turn usually represent some kind of need. In our example, the traveller has the need to get a room and to have wireless internet-access. The hotel's business need (or strategic goal) is to offer such wireless connectivity and thus it has the need towards a certain type of IP access. Further, *AND* and *OR* elements can be used to specify the scenario execution.

2.3 On the Adequacy of Traditional Information Engineering-like Methods for Networked Value Constellations

Methodologies are organized as documented sets of steps and procedures for accomplishing a desired task. Many methodologies (such as IE) include diagramming notations (e.g. entity relationship diagrams) for documenting the results of a certain step. They all start with developing an enterprise model first and then deriving process and data models from the enterprise model. In the literature, the only exception from this planning process is suggested by Gordijn [GA01, GA03], who recommends planers to first focus on the value viewpoint and the conceptualization of a value-base business model and, then, develop the models reflecting the other viewpoints. However, a danger of such multi-perspective

approaches is that a specific viewpoint only implies a partial specification of the system from a particular stakeholder's view. Such a separation of concerns leads to potential inconsistencies between specifications from different perspectives, which may result in a technically non-implementable system. *Consistency checking methods* are currently the only way to ensure successful implementation [ZW05]. Our literature review indicates that, to the best of our knowledge, there is no IS planning methodology that takes value-based requirements as a starting point for determining the IS landscape of a networked value constellation into account. To improve this situation, we propose an enhancement of the existing IE methodology that will meet the following needs of cross-organizational systems planners:

- focusing on multiple enterprises that make up a value web.
- taking requirements from the value viewpoint into account.
- deriving enterprise models from value models.
- taking existing (legacy) systems into account.
- quantitatively analyzing IS architecture options.

Our envisioned IS planning methodology for networked value constellations will enhance existing systems planning practices in four aspects which are summarized in Tab. 1.

Aspect	Traditional IE	IS Planning for Value Networks
business model	enterprise model	value model
organizational structure	single enterprise	networked value constellation
process model	internal business processes	inter-organizational coordination processes
scope	computerize an enterprise from the "scratch"	computerize just the e-business idea (by taking legacy systems into account)

Table 1: Main differences between traditional IE and proposed IS planning methodology for networked value constellations

First, the starting point in cross-organizational systems planning at the strategic level will be the network business model. Its main purpose is to provide a big-picture view of the business (units). This new starting point is an enhancement to traditional IE which puts the enterprise model first. Second, our focus on the value model makes explicit the economic value aspect in network constellations, whereas an enterprise model is focused rather on the companys organizational structure. Third, because traditional approaches focus on single enterprises, they address the needs to model internal business processes only. In contracts to this, our envisioned methodology will keep its focus on multiple enterprises and will allow planners to conceptualize inter-organizational coordination processes that

will operationalize the overlaying value model. Fourth, the new methodology reflects the most recent shifts in the scope of system planning. Traditional IE assumes that planners start from the scratch and completely computerize an enterprise, whereas the adapted methodology would computerize just the e-business idea. This means that, with the new methodology, (i) a systems planning exercise will be focused on the IS landscape that covers only those parts of the business units which are necessarily involved in implementing the e-business idea, and (ii) the planner will not start from a green field, but will consider and possibly re-use legacy systems.

Our analysis makes it clear that the traditional IE approach is not enough to confront the needs of planners in networked value constellations settings. This motivated our efforts to enhance current cross-organizational planning practices. We, however, do not envision to arrive at a completely new methodology, but rather to enhance and adapt existing practices to this new situation. In what follows, we present our initial proposal for an enhanced IE-like methodology and, then, we narrow down the discussion to address one specific question in our proposal, namely, the concern of how to derive a networked enterprise model from a value model.

3 An enhanced Information Engineering-like methodology for Networked Value Constellations

Figure 4 shows our proposed IS Planning methodology. Three major enhancements to traditional IE-like methodologies ([B] in Fig. 4) can be identified:

- $[A]$ deriving (networked) enterprise model from a value model,
- $[C]$ the consideration of legacy systems, and
- $[D]$ economic-driven IS architecture support.

From Value Model to Enterprise Model. We attempt to derive an enterprise model from a value model ([A] in Fig. 4), i.e. we derive data-orientation from value-orientation. This is an important and challenging enhancement in the new methodology, because it enables the straightforward application of the traditional IE-like techniques afterwards ([B] in Fig. 4).

Considering legacy systems. Traditional systems planning approaches have the aim to computerize an uncomputerized enterprise, but these approaches usually end with the identification of natural systems after performing CRUD analysis. Networked value constellations are usually already computerized, i.e. participating partners have an IS landscape. The challenge here ([C] in Fig. 4) is to understand how the proposed natural systems relate to legacy systems. As far as businesses want to reuse their legacy systems, a "gap analysis" has to be performed.

Quantitative decision support. As far as a networked value constellation represents a multi-actor network of independent (profit and loss oriented) business actors, systems realizing coordination among multiple actors might be identified in the previous steps. In order to do so, it must be determined which actor should own and purchase such a system. Such architectural decisions must be made on a rational basis and must be feasible from an economic point of view ([D] in Fig. 4) [Zar06].

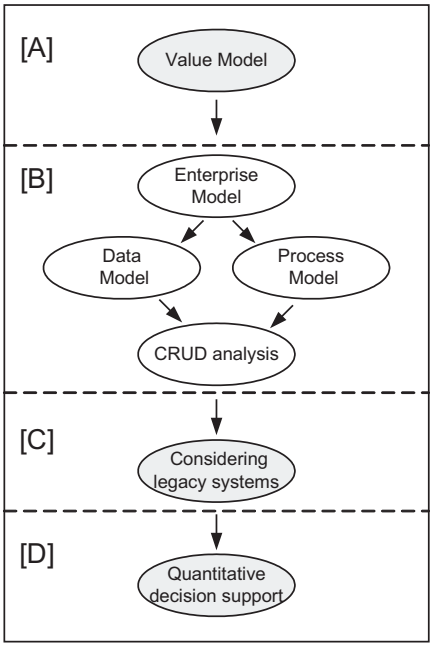


Figure 4: Proposed IS Planning methodology for Networked Value Constellations

In this paper, we concentrate on the first enhancement [A] and show in the next section how to derive an enterprise model from a value model.

4 From Value Model to (Networked) Enterprise Model

Martin describes that the first step in top-down planning is "to create a chart or model of the enterprise" [Mar82, p. 21]. In our approach this is the second step and in order to arrive there, we need to map *value requirements* to such *data requirements*.

Enterprise Models may vary in the degree of detail, depending on how detailed such models are created or on the dimension of the enterprise. This section shows how we get from value models to enterprise charts/models (in this case networked enterprise models). This is achieved by taking (i) e^3 - *value* model as well as the (ii) the textual e-business case de-

scription itself into account, because the latter one might contain information not included in the conceptual value model. In the following we give some guidelines, which are also summarized in Tab. 2.

e^3 - *value* model	Networked enterprise chart
name of value model	highest level in the enterprise chart
value actor and market segments	second level representing the independent business division
value activities	operational activities (functional areas)
value objects and value ports	additional information about operational activities
textual e-business description	additional information about operational activities

Table 2: Deriving a networked enterprise chart from e^3 - *value* modeling artifacts

The highest level of the enterprise chart is determined by the name of the value model itself, in our case the "travelling salesman". This ensures that we are talking about the same e-business constellation. The second level is a straightforward mapping of the different value actors and market segments. These represent the profit-and-loss oriented business units. The value activities these business units perform represent then the next level in the enterprise chart, like e.g. WiFi Access. At this point the difference between value models and process models come into play. The main goal of a value model is to answer *who* is offering *what* to *whom* and expects *what* in return, whereas the process viewpoint is concerned with *how* these value activities are realized [GA03]. Therefore a "value activity forms the linking pin to operational activities" [DG06] and functional areas respectively. Functions like authentication or traffic encryption realize the value activity WiFi Access properly. Therefore functional decomposition of the identified functional area gains in importance. Additionally the value objects themselves as well as the value ports give us indications, e.g. fee/money transfer is realized through some accounting-like function. Last, we can possibly identify from the textual description of the e-business case information not contained in the value model, such as is the case described in [ZW05] where the model shows that fee has to be paid for a certain value object but not how. This can be extracted from the textual description (e.g. cash or bank transfer).

For the example from Fig. 3 the networked enterprise chart is shown in Fig. 5, which in turn enables straightforward use of traditional data-oriented IS planning techniques. Note that the above figure shows only the functions of the e-business idea. The hotel might have additional functions and business areas not listed here, like e.g. food and beverage, personnel, telephone switchboard. An ISP on the other side could next to IP Access also offer services such as registration of domains or act as a content provider. This is the reason why we are talking about a "computerized e-business idea" and not in the classical sense of a "computerized enterprise".

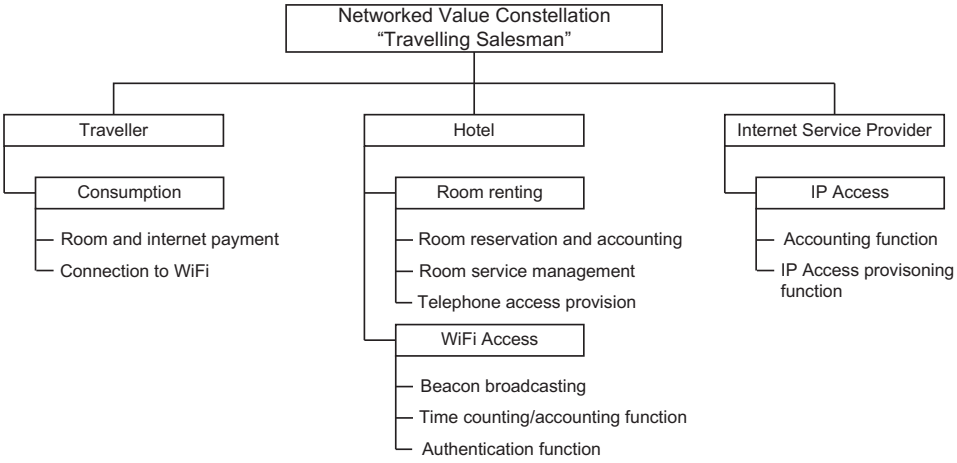


Figure 5: Networked enterprise chart derived from value model of “Travelling Salesman” case

5 Summary

Discussion. In this paper we presented our approach of how to migrate from value-based requirements to data-orientation in order to help systems planners the use of traditional data-oriented systems planning techniques for networked value constellations. It should be noted that the IS landscape, as the outcome, is just representing a global view. In real life two kinds of Networked Value Constellations are conceivable. First, one business actor is so powerful and decides on the architecture so that if smaller actors want to participate in the e-business idea, they need to accept the rules of this business actor [FGP03]. An example of such case is the automotive industry, where small suppliers need to adapt to the given interfaces of big car manufacturers. Second, each actor has the same power in the network, so that the global view, like shown in this paper, can be the driver for the IS landscape.

Future Work. Our current approach rests on experiences published in the literature that indicated why existing methods come short when planning cross-organizational systems. A limitation of this approach is that we did not determine it based on context attributes in a real-life case study. However, informal conversations with business partners from our project indicate that it makes sense to try it out in their planning settings. We, therefore, plan in the future a series of case studies of different size in the Netherlands in which we hope to gain more knowledge on the organizational context characteristics that favour the application of our methodology and the ones that make it weak. This, of course, includes further research and refinement of the presented steps. We also plan to refine and extend existing recommendations for good systems planning practices from earlier published experiences [FGP03].

References

[BH06] S. Berman and J. Haggan. How technology-driven business strategy can spur innovation and growth. *Journal of Strategy and Leadership*, 34(2):28–34, 2006.

[Bin96] C. Binkert. Eine systematische Vorgehensweise für Anwendungsprojekte. In W. Brenner, C. Binkert, and M. Lehmann-Kahler, editors, *Information Engineering in der Praxis*. Campus Verlag, Frankfurt, 1996. (In German).

[CST06] Y. Chan, R. Sabhervawal, and J. Thatcher. Antecendents and Outcomes of Strategic IS Alignment: an Imperical Investigation. *IEEE Transactions on Engineering Management*, 53(1):27–47, 2006.

[DG06] Z. Derzsi and J. Gordijn. A Framework for Business/IT Alignment in Networked Value Constellations. In *Proceedings of Workshops and Doctoral Consortium of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06), Luxembourg*, 2006.

[FGP03] P. Finnegan, R. Galliers, and P. Powell. Systems Planning in Busines-to-Business Electronic Commerce Environments. *Journal of Information Technology and Management*, 4(2-3):183–198, 2003.

[Fin89] C. Finkelstein. *An Introduction to Information Engineering - From Strategic Planning to Information Systems*. Addison-Wesley, Sydney, 1989.

[GA01] J. Gordijn and H. Akkermans. Designing and Evaluating E-Business Models. *IEEE Intelligent Systems*, 16(4):11–17, July/August 2001.

[GA03] J. Gordijn and H. Akkermans. Value-based requirements engineering: exploring innovative e-commerce ideas. *Requirements Engineering*, 8(2):114–134, July 2003.

[Hay03] D. Hay. *Requirements Analysis*. Pearson Education, New Jersey, 2003.

[IA05] I. Ishak and R. Alias. Designing a Strategic Information Systems Planning Methodology for Malaysian Institutes of Higher Learning (ISP-IPTA). *Issues in Information Systems*, VI(1):325–331, 2005.

[LM89] A. Lederer and A. Mendelow. Coordination of Information Systems Plans with Business Plans. *Journal of Management Information Systems*, 6(2):5–19, 1989.

[Mar82] J. Martin. *Strategic Data-Planning Methodologies*. Prentice-Hall, New Jersey, 1982.

[Mar89] J. Martin. *Information Engineering*. Prentice Hall, New Jersey, 1989. Three Volumes.

[PR01] S. Pant and T. Ravichandran. A framework for information systems planning for e-business. *Logistics Information Management*, 14(1/2):85–98, 2001.

[PS95] S. Pant and C. Su. Strategic Information Systems Planning: A Review, 1995.

[RB00] B. Reich and I. Benbasat. Factors that Influence the Social Dimension of Alignment between Business and Information Technology Objectives. *MIS Quarterly*, 24(1):81–113, March 2000.

[Tap06] R. Santana Tapia. What is a Networked Business? Technical report, University of Twente, 2006.

[TF97] R. Tagg and C. Freyberg. *Designing Distributed And Cooperative Information Systems*. International Thomson Computer Press, 1997.

[WBFG03] R. Wieringa, H. Blanken, M. Fokkinga, and P. Grefen. Aligning Application Architecture to the Business Context. In *Conference on Advanced Information System Engineering (CAiSE 03)*, pages 209–225. Springer, 2003. LNCS 2681.

[WG98] A. Wong and F. Gregory. The Inapplicability of Western Models to Information Technology Development in Chinese Companies - The Case of the Hong Kong Newspaper Industry. In *Proceedings of the Fifth IFIP WG 9.4 Working Conference: Implementation and Evaluation of Information Systems in Developing Countries*, Bangkok, February 1998.

[WGvE05] R. Wieringa, J. Gordijn, and P. van Eck. Value-Based Business-IT Alignment in Networked Constellations of Enterprises. In *Proc. 1st International Workshop on Requirements Engineering for Business Needs and IT Alignment (REBNITA 2005)*, 2005.

[Zar06] N. Zarvic. Value-Based Requirements Engineering and IS Architecture Design Support for Cross-Organizational Environments. In *Proceedings of the Doctoral Symposium of the 14th IEEE International Requirements Engineering Conference (accepted)*, 2006.

[ZW05] Z. Zlatev and A. Wombacher. Consistency between e^3 - *value* Models and Activity Diagrams. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, pages 520–538, Cyprus, November 2005.

On Design Principles for Realizing Adaptive Service Flows with BPEL

Manfred Reichert[1] and Stefanie Rinderle[2]

[1]Information Systems Group, University of Twente, The Netherlands
m.u.reichert@utwente.nl
[2]Dept. Databases and Information Systems, Ulm University, Germany
stefanie.rinderle@uni-ulm.de

Abstract: Web service technology offers a promising approach for realizing enterprise-wide and cross-organizational business applications. With the *Business Process Execution Language for Web Services* (BPEL, also known as WS-BPEL or BPEL4WS) a powerful language for the process-oriented composition and orchestration of Web services exists. However, BPEL flow specifications tend to be too complex, and current BPEL engines do not provide the flexibility needed to cover the broad spectrum of business processes we have to deal with in practice. Neither ad-hoc deviations form the pre-specified BPEL schema (e.g., to add or move activities for single flow instances) nor the propagation of BPEL schema changes to a collection of flow instances have been supported by BPEL engines yet. This limits their applicability to only well-structured, rigid service flows. In this paper we address fundamental issues, which arise when enriching BPEL engines with dynamic change capabilities. We focus on the question how such changes can be realized in a correct and consistent manner and which prerequisites must be met in this context. In particular, we restrict BPEL to a reasonable subset of language elements in order to decide on these fundamental questions. By offering flexibility and adaptability in a controlled and reliable way the so promising Web Service technology will broaden its application scope significantly.

1 Introduction

Today there is a high need for active coordination of the various, distributed tasks necessary to perform enterprise-wide business applications. Service-oriented architectures offer a promising approach in this context [WCL+05]. Usually, they provide a framework for specifying, implementing, and registering application services as well as for composing them in a process-oriented manner. The latter enables stateful interactions among the services and provides the basis for reliable process orchestration. In this context, the *Business Process Execution Language for Web Services* (BPEL, also known as WS-BPEL or BPEL4WS) has emerged as de-facto standard for implementing processes based on (Web) services [vDt+05, OAS06]. Systems like Oracle BPEL Process Manager, IBM WebSphere Process Server, and Microsoft BizTalk Server support BPEL, thus illustrating the practical relevance of this process description language. Though intended as a language for con-

necting web services, it can be expected that in near future a wide variety of process-aware information systems (PAIS) will be realized using BPEL [vDt+05].

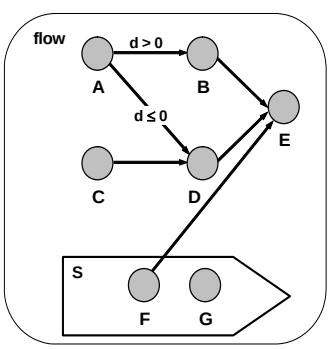
Whilst BPEL is an expressive language, it is by far too complex to realize more advanced process support functions. In particular, the aforementioned BPEL engines do not provide the flexibility and dynamism needed to cover the broad spectrum of business processes we can find in todays organizations. Neither ad-hoc deviations from the pre-planned service flow (e.g., to add, shift or delete activities in order to deal with an exceptional situation) nor the propagation of a business process change to a collection of flow instances have been supported yet. This, in turn, requires from users to bypass the system in exceptional cases or prohibits already running flow instances from being adapted to an optimized business process. In many domains, however, PAIS will not be accepted by users when rigidity comes with them. Creating service-oriented applications without a vision for adaptive process management, therefore, is shortsighted and expensive. Indeed, insufficient flexibility has been a primary reason why workflow technology failed in many process automation projects in the past. Obviously, similar problems will result for BPEL-based process engines if dynamic flow changes are prohibited a priori by a bad language design.

Barriers on the way towards adaptive BPEL flows are the complexity of BPEL and the missing formal semantics of this language. Though several approaches exist in this context (e.g., [Mar05, OvB+05, RRD04a]), current formalizations and verification methods are incomplete (i.e., focussing on selected BPEL language elements only) and are not standardized. In particular, one will not be able to reason about the correctness of dynamic flow changes if there exist no formal basis and no mechanism for reasoning about correctness issues. In this paper we sketch how existing concepts from the area of adaptive processes (see [RRD04b] for an overview) can be applied to restrict and configure the BPEL language in a way such that it becomes a candidate for realizing adaptive flows. Due to lack of space we mainly focus on design principles and omit formal issues.

Section 2 summarizes background information related to the modeling of BPEL flows. Based on this, Section 3 introduces guidelines for a corresponding flow execution model. Using this model, Section 4 deals with the question how to check whether a given BPEL flow instance is compliant with a modified flow schema or not. Section 5 copes with the state adaptations becoming necessary in this context. Finally, Section 6 discusses related work and Section 7 concludes with a summary and an outlook.

2 Background Information

BPEL uses an expressive meta model for creating XML-based descriptions of business processes based on the interactions between the process and its partners. In the following we omit XML specifications and use graphical illustrations instead. In a BPEL flow, the interaction with each partner occurs through web service interfaces. Process activities can invoke service operations of partners synchronously, or they may receive messages from service invocations of partners and reply to them asynchronously later. BPEL provides a variety of possibilities to describe the desired flow logic. For example, Fig. 1 shows a




```
...
<flow>
  <links>
    <linkname = "L_AB"/>
    <linkname = "L_AD"/>
    <linkname = "L_FE"/>
    ...
  </links>
  activity A
    <source linkName = "L_AB" transitionCondition="d>0"/>
    <source linkName = "L_AD" transitionCondition="d≤0"/>
  activity B
    <target linkName = "L_AB"/>
    <source linkName = "L_BE"/>
  ...
  activity E
    joinCondition = "(L_BE OR L_DE) AND L_FE"
    <target linkName = "L_BE"/>
    <target linkName = "L_DE"/>
    <target linkName = "L_FE"/>
  <sequence name = "S">
     activity F
       <source linkName = "L_FE"/>
     activity G
  </sequence>
</flow>
```

Figure 1: Example of a BPEL flow schema (modeled in a graph-like fashion)

BPEL flow schema which has been modeled in a graph-like fashion, i.e., the flow logic is described as a network of activity nodes and (control) links. For such a graph-based BPEL flow, a variety of configuration facilities exist, like the assignment of transition conditions to (control) links (i.e., predicates on flow variables), and the definition of activity join conditions. However, the flow from Fig. 1 can be also modeled in a more block-oriented fashion as depicted in Fig. 2. This flow schema is based on structured activities (sequence, flow, switch) and two links which synchronize activities from parallel branches. Generally, structured activities can be nested. In our example, the switch activity is contained in a sequence activity, which itself is surrounded by a flow activity (representing parallelism).

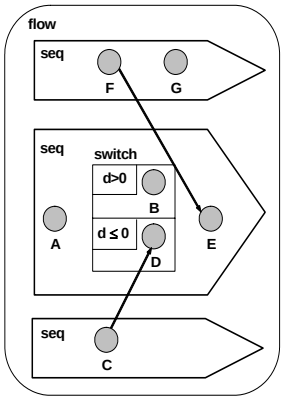


Figure 2: BPEL flow from Fig. 1 modeled in a block-like fashion

Generally, it is possible to mix both modeling styles by having links crossing the boundaries of structured activities. On the one hand BPEL provides high expressiveness and many ways to express the same thing; on the other hand this aggravates flow modeling, analysis and verification (see [RRD04a] for details). In order to reduce this complexity and

to be able to reason about correctness properties of BPEL flow specifications, in [RRD04a] we have introduced a classification for BPEL flow schemes. In this work, we have also shown that block-structured modeling, together with the controlled use of links, offers advantages when compared to the graph-like modeling style. Note that such considerations become necessary when reasoning about the correctness of flow schema changes. However, for the rest of this paper we abstract from issues related to schema correctness, and assume that changes of BPEL flow schemes are handled in a correct manner. Instead we focus on the propagation of such flow schema changes to running flow instances.

3 Designing an Execution Model for BPEL Flows

A prerequisite for dynamically adapting the structure and state of BPEL flow instances is the provision of a formal semantics. It constitutes the basis for both the correct execution of BPEL flow instances and the correct adaptation of their states when applying dynamic changes. Regarding flow instance changes it must be also possible to quickly decide whether a given flow instance or a collection of flow instances is compliant with the modified flow schema, and may therefore be migrated to it. In the following we discuss guidelines for the design of such an (adaptive) flow execution model for BPEL. Our considerations are based on a previous survey on formal models for adaptive process management (cf. [RRD04b]). Due to lack of space we focus on fundamental issues and exclude more advanced BPEL process patterns (e.g., scopes and exception handlers).

We distinguish between different states a single BPEL activity (i.e., a single process step) may run through. Initially, an activity has state *NotActivated*. It will change to *Activated* (i.e., the activity becomes enabled) if all preconditions for executing the activity are met. When starting it (e.g., by invoking a Web Service operation or by receiving a message from a partner) we obtain activity status *Running*. At successful termination activity status passes to *Completed* whereas processing failures result in status *Failed*. Finally, status *Disabled* represents activities of non-selected paths (e.g., activities whose execution has been skipped due to a dead path elimination or which belong to a non-selected branch of a switch block). In addition, for each instance I an execution history $H_I =< e_0, ..., e_k >$ is kept. It logs events related to the start and termination of flow activities.

A flow execution model for BPEL must provide rules for initializing activity states when creating new flow instances and for adapting these states when activities are started or completed. More precisely, these rules must enable the runtime system to decide in which flow states a certain activity can be activated, and to determine the correct follow-up state of the flow instance when activities are completed. To achieve this, for basic as well as structured activities a well-defined operational semantics is required. For example, an adequate representation of both activity and link states – also denoted as activity / link markings in the following – is needed. Generally, different options exist in this context. The first one uses only one type of control token passing through each flow instance (True-Tokens). An alternative is to use two types of tokens (True- and False-Tokens). Simplistically, True-Tokens trigger activities that are to be executed next and False-Tokens describe disabled activities. Flow description formalisms which solely use True-Tokens include Petri-Nets [RRD04b].

Approaches which additionally use False-Tokens usually preserve the markings of already passed regions (except for loop backs) and explicitly mark disabled activities. These approaches can be further divided according to the way they represent the tokens. One option is to gain them from execution histories (which log events like activity start / completion. Alternatively, (model-inherent) activity markings, representing a consolidated view on execution logs, can be used [RRD04b].

As discussed in [RRD04b] the use of a flow execution model with True-/False-Tokens offers the best perspectives with respect to dynamic process changes. A corresponding example is depicted in Fig. 3. As can be easily seen from this figure, activity markings do not only indicate whether an activity is currently activated or disabled, but also provide a consolidated view on previous instance execution; i.e., on the execution history of the respective instance. As we show in the following this is exactly what we need to efficiently check compliance of a flow instance with a modified BPEL flow schema and to automatically adapt markings of this instance when migrating it to the new schema version.

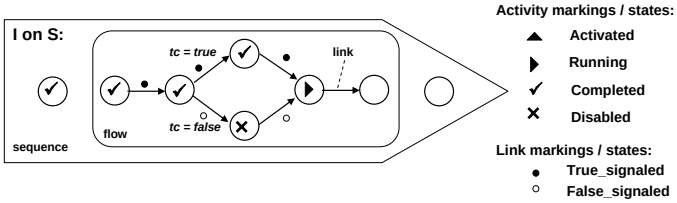


Figure 3: BPEL flow instance with activity markings (True-/False semantics)

In principle, BPEL is coherent with this way of representing activity states. However, in our context we need a more formal definition of the underlying operational semantics as currently provided by the BPEL specification. In any case, BPEL shows a good potential for satisfying the following basic properties needed for the support of adaptive flows:

1. The flow execution model must offer different kinds of activity markings to distinguish between executed and disabled flow paths (True-/False-Semantics).
2. Internal activity states must be reflected by the flow execution model. For example, it must be possible to distinguish between enabled activities and already started ones. This is crucial for checking compliance of a flow instance with a modified schema.
3. The markings of already passed regions should be (logically) preserved (except loop backs); they provide a consolidated view on the previous instance execution.
4. The flow execution model must provide precise execution and marking rules, which state under which marking an activity can be executed and which new marking results when completing it. Such a flow execution model should be based on a graph-based representation of BPEL schemes.

Fig. 3 shows a simple BPEL flow instance – a sequence block with an embedded flow activity as well as embedded basic activities – together with its current activity and link

markings. Note that this representation already satisfies the execution properties 1–3 mentioned above. At this point it is important to mention that activity/link markings only represent a logical view on the respective flow instance. How these markings are internally represented, how they are physically stored, and which optimizations are conceivable in this context depends on the concrete BPEL implementation and is outside the scope of this paper. The same applies to the concrete set of marking and execution rules.

4 A Correctness Criterion for Changing BPEL Flow Instances

We now deal with the challenge to propagate changes of a BPEL flow schema to corresponding flow instances. The ambition must be to preserve correctness of these instances when dynamic changes are carried out. We need adequate criteria to decide whether an instance I is compliant with a changed BPEL schema; i.e., whether the change can be correctly propagated to I without causing errors (like deadlocks or improperly invoked services). For this we need an adequate correctness criterion which does not needlessly exclude a flow instance from being migrated to a new schema version. Furthermore, it must be ensured that correctness can be efficiently checked by the BPEL engine.

4.1 Preliminaries

We assume that BPEL schema changes are correctly accomplished. However, this is not sufficient to guarantee a consistent execution behavior of flow instances when migrating them to the new BPEL schema version. Additionally, we must consider instance states. To illustrate the problems resulting from an uncontrolled migration consider the example from Fig. 4. The BPEL schema S from Fig. 4a) represents a *sequence* consisting of activities A, B and C. Assume that S is correctly transformed into S' by adding activities X and Y as well as a data dependency between them – Activity Y reads variable d which is written by the preceding activity X (cf. Fig. 4a). Assume further that these changes are propagated in an uncontrolled way to the flow instances from Fig. 4c) (currently running on S). Concerning instance I_1 no problem would occur since its execution has not yet entered the region affected by the change (activity A is still running). Uncontrolled migration of instance I_2, however, would cause malfunctions. First, an inconsistent state would result – B is already running though its (new) predecessor X (on S) has not been completed. Second, the newly inserted activity Y might be invoked though variable d has not been written (by X). For flow instance I_3 migration would be possible. When doing so, however, activation of B has to be undone and the added activity X be activated instead. – In the following we do not distinguish between changes of single flow instances (e.g., to deal with exceptions) and adaptations of a collection of flow instances (e.g., to deal with a process type change). Instead we focus on fundamental issues related to dynamic changes of BPEL flow instances and sketch how they can be addressed.

Our previous example has demonstrated that the applicability of a dynamic flow change

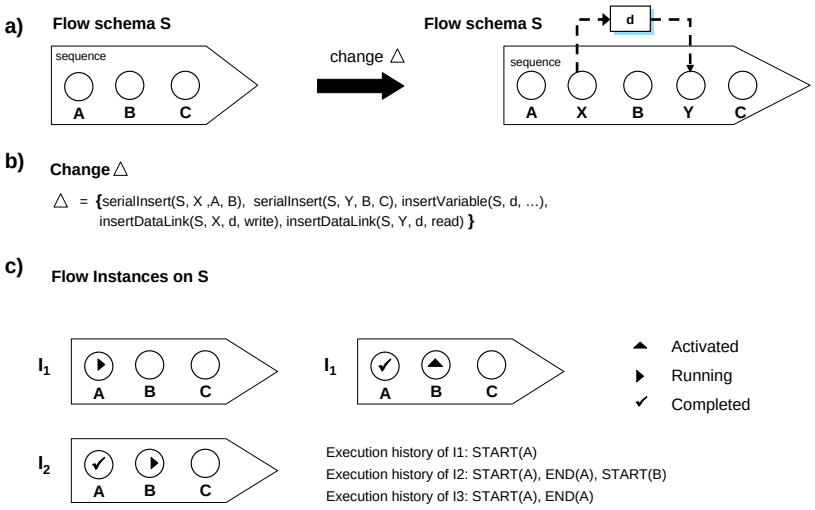


Figure 4: BPEL flow schema and related flow instances

depends on the current flow state as well as on the kind of changes being applied. Let I be a flow instance with BPEL schema S; assume that S is transformed into another ”correct” BPEL schema S' by applying change Δ. Then two fundamental issues arise:

1. Can Δ be correctly propagated to flow instance I, i.e., without causing errors and inconsistencies? If yes, we denote I as compliant with respect to S'.

2. Assuming I is compliant with S', how can we smoothly migrate it to S' such that its further execution can be based on this new BPEL schema? Which state adaptations (e.g., enabling/disabling of activities) become necessary in this context?

While the first issue concerns pre-conditions on the current state of I, the second one is related to post-conditions that must be satisfied after propagating the change. In any case we have to provide an efficient solution which enables automatic and correct compliance checks as well as automated instance migrations.

4.2 A General Correctness Criterion

In [RRD04c] we have introduced a general correctness criterion, which is independent of the used process description formalism and which allows us to argue about the correctness of dynamic flow changes. In more detail, this criterion enables us to decide whether a given flow instance can be migrated to the modified schema or not (so-called *compliance*). In addition, we provide automated procedures to determine the correct follow-up state of compliant flow instance resulting after their migration.

Simplistically, flow instance I would be compliant with a (modified) flow schema S' if I could have been executed according to S' as well, and would have produced the same effects on flow state and variables. Trivially, this will be always the case if I has not yet entered the region affected by the flow schema change. Generally, we need information about previous flow execution to decide on this property and to determine correct follow-up states for the (compliant) flow instances to be migrated. For this purpose, we can make use of the execution history which is kept for each flow instance (cf. Fig. 4c) independent of the applied flow description language. Usually, this history logs events related to the start / completion of flow activities. Obviously, flow instance I with history H_I will be *compliant* with S' (and therefore be able to *migrate* to this schema) if H_I can be produced on S' as well. Consequently we then obtain a correct new state of flow instance I on schema S' by replaying all events from H_I on S' in chronological order. Taking our example from Fig. 4 this holds for I_1 and I_3, but does not apply to I_2. Furthermore, when replaying H_I on S', we obtain the new flow states as sketched above. Note that the aforementioned criteria are independent of the underlying flow description language.

The described criterion is still too restrictive to serve as general correctness principle. Particularly with respect to loops[1] it may needlessly exclude flow instances from being migrated to a modified schema. As an example take a flow instance I for which a loop block is in its 2nd iteration. Assume that the flow schema shall be modified by adding activities to a non-entered region from this loop block. Taking the above criterion this change will be not allowed if the first loop iteration of I is not compliant with the resulting flow schema; i.e., H_I could not be completely produced on the modified schema. However, excluding such flow instances from migrations is not in accordance with practice. To cope with this we relax the above criterion by (logically) discarding those history entries from the execution history produced within another loop iteration than the last (completed loops) or current one (running loops). We denote this reduced view on execution history H_I as *reduced execution history* RH_I. Based on this we can apply the following fundamental correctness principle for dynamic changes of BPEL flow instances:

Axiom 1 (Dynamic Change Correctness) *Let I be a BPEL flow instance on schema S with execution history H_I and reduced execution history RH_I. Assume further that S is transformed into a correct schema S' by applying change Δ. Then:*

1. *Δ can be correctly propagated to BPEL flow instance I iff RH_I can be produced on S' as well – for this case I is said to be compliant with S'.*
2. *When propagating Δ to a flow instance I, which is compliant with S', the correct state of I on S' (e.g., activity markings) can be obtained by replaying RH_I on S'.*

4.3 Rules for Automatically and Efficiently Checking Compliance

The challenge is how to efficiently guarantee compliance and how to determine the correct new state of compliant flow instances (according to Axiom 1) in BPEL. Certainly, it

[1] Loops can be modeled in BPEL by means of a specific block element.

would be no good idea to access the whole (reduced) execution history and to try to replay it on the modified BPEL flow schema. This may cause a performance penalty due to the large number of instances to be treated. In the following we present optimized rules and procedures for BPEL flows in order to ensure dynamic change correctness according to Axiom 1. For selected change operations we examplarily sketch compliance conditions which can be efficiently checked by an adaptive BPEL flow engine. We apply the following design principles: (1) We consider change semantics and change context in order to precisely specify which state information is needed for checking compliance. (2) We make use of the dynamic model properties sketched in Section 3. In particular, the derivation of compliance rules benefits from model-inherent activity markings which already provide a consolidated view on the (reduced) history of a flow instance.

In the following let S be an executable BPEL flow schema and let I be a flow instance on S with marking MS = (NS, ES)[2]. Assume that S is correctly transformed into another BPEL flow schema S' by applying change Δ[3]. For selected changes Δ, Fig. 5 shows the formal conditions under which we can ensure compliance of I with the modified BPEL flow schema S'. More precisely, one can formally prove that flow instance I is compliant with flow schema $S' = S + \Delta$ (according to Axiom 1) if the compliance rule related to Δ is satisfied. Note that none of the change operations depicted in Fig. 5 requires access to the (complete) execution history. Generally, this cannot always be achieved. Concerning the insertion of a new link, for example, in certain cases compliance can be guaranteed even if the target activity of this link has been started, completed, or disabled. To decide on this, however, additional information from execution logs is needed.

Change Operation Δ on schema S...	**... and Related Compliance Condition**
addActivity(S, act, Preds, Succs) *Inserts activity act between node sets Preds and Succs; i.e., each node of Preds will become a predecessor and each node of Succs a successor of act.*	[∀ n ∈Preds: NS(n) = Disabled] ∨ [∀ n ∈Succs: (NS(n) ∈ {NotActivated, Activated}) ∨ (NS(n) = Disabled ∧ ∀ m ∈ succs(S,n): NS(m) ∈ {NotActivated, Activated, Disabled})] *(succs(S,n) denotes the set of direct and indirect successors of n in flow schema S)*
deleteActivity(S, act)	NS(act) ∈ {NotActivated, Activated, Disabled}
addVariable(S, var, ...)	no condition
addDataLink(S, act, var, read) *Adds a read data link to schema S; i.e., activity act will have read access on variable var.*	NS(act) ∈ {NotActivated, Activated, Disabled}
addDataLink(S, act, var, write) *Adds a write data link to schema S; i.e., activity act will have write access on variable var.*	NS(act) ≠ COMPLETED

Figure 5: Examples of compliance rules for dynamic changes of BPEL flows

We exemplarily describe the compliance condition of change operation addActivity. Re-

[2]To each activity of the respective flow instance NS assigns one of the markings *NotActivated, Activated, Completed, Disabled or Failed,* and to each link of the respective flow instance ES assigns one of the states *NotSignaled, TrueSignaled or FalseSignaled*

[3]What BPEL schema correctness means in our context has been discussed in [RRD04a]

garding the insertion of an activity act between two node sets $Preds$ and $Succs$ compliance can be guaranteed if all activities from $Succs$ possess one of the markings *Activated* or *NotActivated*. In this case, none of the successors of act has yet written an entry into the execution history H_I. A simple example is depicted in Fig. 6 where activities X and Y as well as a data dependency between them are added. Furthermore, compliance can be also ensured if all activities from $Preds$ are marked as *Disabled*. Then the added activity will be disabled as well, i.e., its insertion will have no effect on compliance. Finally, activity act may be added as predecessor of a disabled activity provided that none of the successors of this activity has a marking other than *NotActivated, Activated, or Disabled*.

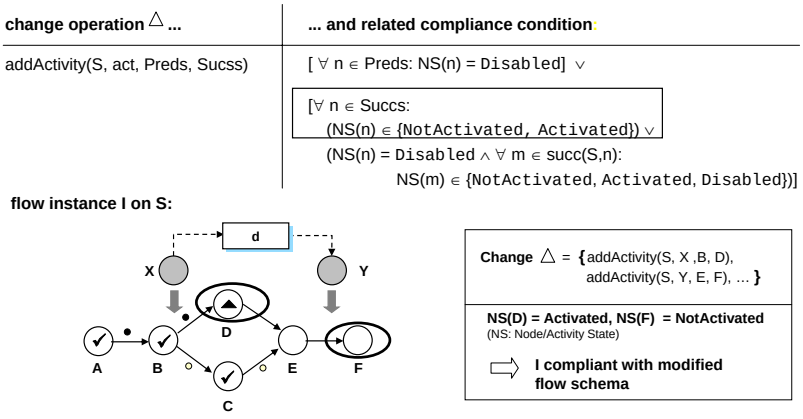


Figure 6: Checking compliance when inserting new activities

Fig. 5 shows compliance conditions for several other change operations (e.g., deletion of activities, insertion of data links). In a similar way, compliance conditions for other change primitives can be derived (e.g., insertion/deletion of structured activities, update of a transition/join condition, etc.). Altogether we can state that compliance (cf. Axiom 1) very often can be decided on basis of current activity markings; i.e., the engine must not explicitly check the producibility of execution histories on the modified schema. In order to come to a complete solution, corresponding compliance conditions must be derived for all basic change operations applicable on a BPEL schema. Based on them, one can develop high-level change operations and related compliance rules. Since the latter can be derived by merging compliance conditions of the applied change operations and by discarding unnecessary expressions (e.g., conditions on nodes not present in the original flow schema) we do not consider them in this paper. As mentioned, Fig. 5 only presents the compliance conditions for selected change primitives. One remaining task is to provide a complete set of change operations on BPEL schemes and to derive respective compliance conditions for them. One challenge in this context is the high expressiveness of the BPEL4WS language.

5 Automatically Adapting BPEL Flows After Changes

We exemplarily show how compliant flow instances can be migrated to a changed flow schema. One problem to be solved is the correct and efficient adaptation of activity and link markings. How extensive marking adaptations turn out for an instance I to be migrated depends on the kind and scope of the change. Except initialization of newly added activities and links, for example, no adaptations will become necessary if execution of I has not yet entered the change region. In other cases extensive marking adaptations may be required. An activity X, which is enclosed by a flow activity, may have to be deactivated if new links are inserted with X as target activity. Conversely, an added activity Y will have to be immediately activated or disabled if all predecessors of Y possess a final marking (*Completed, Disabled*). As shown in Fig. 7, it may even become necessary to reset disabled activities in their state when adding an activity.

According to Axiom 1,(2) the marking of a migrated flow instance must be the same as it could be obtained when replaying the respective (reduced) history on the modified flow schema. Following the sketched approach, however, it does not become necessary to access execution logs. Instead the necessary adaptations of activity/link markings can be automatically and efficiently done by (re-)evaluating the markings of those activities/links, which constitute the context of a change region. Regarding activity insertion, for example, one must re-evaluate the marking of the added activity itself as well as of its successors. With respect to insertion of a new link, the marking of this link and its target activity must be evaluated. We omit a presentation of algorithmic details and discuss the basic principles along two simple examples instead.

Example 1 (Adding a new activity): A first example is depicted in Fig. 7. Control flow is defined by a network of links. Activities A and D have been completed, whereas activities B and C have been disabled due to a dead path elimination. Let us assume that activity X shall be inserted as successor of A and as predecessor of C. Though C is disabled, this change is allowed since there is no successor of C which has been started yet. Thus the given flow instance is compliant with the modified BPEL schema. When migrating it to this schema, several marking adaptations become necessary to correctly proceed with the flow of control. At first, the incoming link of activity X is evaluated to True_Signaled (since A has marking Completed and the inserted link has no transition condition). This, in turn, leads to activation of X. Since the outgoing link of X cannot be signaled at the moment, the marking of its target activity C (and of C's successor E) is reset to NotActivated. – This simple example illustrates that changes cannot only be applied to the not yet entered regions of a BPEL flow schema. Very helpful for efficiently deciding on the compliance of a flow instance with a modified BPEL schema and for adapting markings is the distinction between completed and disabled activities (True-/False-Semantics) as well as the preservation of markings of already passed regions.

Example 2 (Changing Activity Orders): Consider Fig. 8. Let us assume that, at a certain point in time, a BPEL flow instance looks like the one depicted in Fig. 8a): Activities A and B have been completed and activity C is currently activated (i.e., C is read to start). Normally, A, B, C, D, and E have to be executed in sequence. Assume that an exception occurs which makes it necessary to immediately perform D (which is currently in state

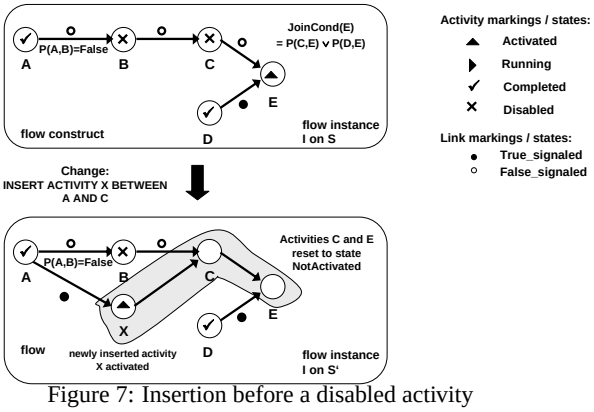


Figure 7: Insertion before a disabled activity

NotActivated) while maintaining the order of all other activities. At first, data dependencies for D are checked; D is immediately executable, in principle, because it reads flow variable d which has been already written by preceding activity B. In order to make it possible that D can be immediately started, it must no longer be a successor of C (because it would have to wait for completion of C). Instead it must be arranged in parallel to C. This can be achieved, for example, by inserting a flow activity (after B) and by placing C and D within it. Finally, the marking of the shifted activity D is re-evaluated leading to the activation of D (the predecessor B of the respective flow activity is completed). Fig. 8c) illustrates how the flow instance looks like after this change.

As a second order changing example, once again take Fig. 8a) and assume that the execution order of activities D and C shall be swapped. This change is allowed since activity C has not been started yet, i.e., the flow instance is compliant with the resulting BPEL schema. (If activity C had been already started or completed, the flow instance would be not compliant with the intended schema (completion of D before starting C); i.e., the change would be not possible.) Finally, markings of activities C and D, which constitute the change region, are re-evaluated leading to activation of D and deactivation of C.

6 Related Work

The introduction of service oriented architectures has increased the flexibility of process-centered software architectures. For example, late binding of service implementations to service specifications enables late modeling of sub-processes as described in [Han97, HJH96]. However, this kind of flexibility is orthogonal to structural flow adaptations.

There are only few approaches which systematically deal with modeling and verification issues related to BPEL control flow specifications [BK03]. As shown in [KMW03, WvDt02], flaws such as service flows which run into deadlocks or service operations which are invoked with missing input data can be avoided by using BPEL as flow de-

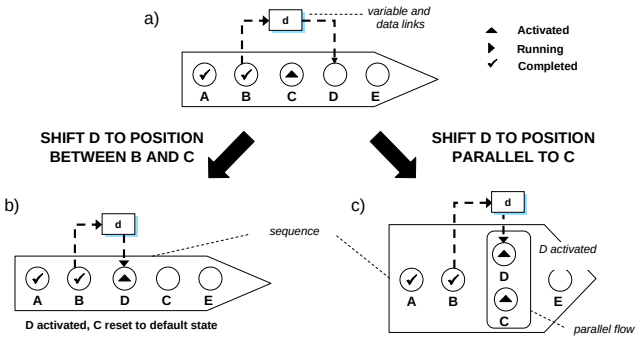


Figure 8: Changing activity order of an in-progress flow instance

scription language. Reason is that it allows to model the flow logic of business processes separately and independently from the implementation of the used Web Services.

In addition to static correctness issues lots of research has been spent on the dynamic adaptation of business processes in general ranging from ad-hoc modifications of single process instances [RD98] to process schema evolution [vB02, CCPP98, RRD04b, Wes01]. However, so far, it has not been investigated whether or not – and if yes how – these results can be applied to BPEL flows as well. In this paper we have not developed new concepts for adaptive processes, but have shown that existing approaches can be applied to BPEL as well. However, this necessitates a more formal and profound basis for this language.

There is ongoing work on the evolution of process choreographies (in addition to process orchestrations such as in this paper) [RWR06]. Focus is put on the correctness of a process choreography after changing the private process of one of the partners involved. For this the authors provide criteria based on which the correct propagation of private process changes to public and private processes of the other partners can be ensured.

7 Summary

We have proposed an approach for adapting BPEL flows based on guidelines for modeling and execution. Correctness requirements are met by a general correctness criterion based on execution logs. In order to verify correctness in an efficient manner (e.g., at the presence of thousands of running BPEL flows) we have elaborated state conditions which are easily to check. Furthermore, we have shown how to adapt BPEL flow states after applying a change. Due to lack of space we have omitted formal details and algorithms, and have presented examples instead. This work has focused on process orchestrations. However, in future work, we aim at tackling change and evolution of process choreographies as well. Challenges in this context comprise the propagation of private flow changes to the public and private flows of partners in order to preserve correctness of the choreography.

References

[BK03] F. Breugel and M. Koshkina. Verification of business processes for web services. Technical Report CS-2003-11, York University, Ontario, CA, 2003.

[CCPP98] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow Evolution. *Data and Knowledge Engineering*, 24(3):211–238, 1998.

[Han97] Y. Han. *Software Infrastructure for Configurable Workflow Systems.* Dissertation, Fachbereich Informatik, TU Berlin, 1997.

[HJH96] T. Herrmann and K. Just-Hahn. Organizational Learning with Flexible Workflow Management Systems. *SIGOIS Bulletin*, 17(3):54–57, 1996.

[KMW03] R. Khalaf, N. Mukhi, and S. Weerawarana. Service-oriented composition in BPEL4WS. In *Proc. WWW'03*, Budapest, 2003.

[Mar05] A. Martens. Analyzing Web Service based Business Processes. In *Proc. Int'l Conf. on Fundamental Appr. to Software Eng. (FASE'05)*, LNCS 3442, Edinburgh, April 2005.

[OAS06] OASIS. *Web Services Business Process Execution Language Version 2.0 - Committee Draft*, 2006. http://www.ibm.com/developerworks/library/ws-bpel/.

[OvB+05] C. Ouyang, W.M.P. van der Aalst, S. Breutel, M. Dumas, A.H.M. ter Hofstede, and H.M.W. H.M.W. Verbeek. Formal semantics and analysis of control flow in WS-BPEL. Technical Report Ext. rep. 05-13, BPM Center Report, Eindhoven, 2005.

[RD98] M. Reichert and P. Dadam. ADEPT$_{flex}$ - Supporting Dynamic Changes of Workflows Without Losing Control. *JIIS*, 10(2):93–129, 1998.

[RRD04a] M. Reichert, S. Rinderle, and P. Dadam. On the Modeling of Correct Service Flows with BPEL4WS. In *Proc. EMISA'04*, pages 117–128, Luxembourg, 2004.

[RRD04b] S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria for Dynamic Changes in Workflow Systems – A Survey. *Data and Knowledge Engineering, Special Issue on Advances in Business Process Management*, 50(1):9–34, 2004.

[RRD04c] S. Rinderle, M. Reichert, and P. Dadam. Flexible Support Of Team Processes By Adaptive Workflow Systems. *Distributed and Parallel Databases*, 16(1):91–116, 2004.

[RWR06] S. Rinderle, A. Wombacher, and M. Reichert. On the Controlled Evolution of Process Choreographies. In *Proc. 22nd Int. Conf. on Data Engineering*, Atlanta, 2006.

[vB02] W.M.P v.d. Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoret. Comp. Science*, 270(1-2):125–203, 2002.

[vDt+05] W.M.P. v.d. Aalst, M. Dumas, A.H.M. ter Hofstede, E. Verbeek, and P. Wohed. Life after BPEL. In *Proc. WS-FM 2005*, LNCS 3670, pages 35–50, 2005.

[WCL+05] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Toney Storey, and Donald Ferguson. *Web Services Platform Architecture*. Prentice Hall, 2005.

[Wes01] M. Weske. Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In *Proc. HICSS-34*, 2001.

[WvDt02] P. Wohed, W.M.P. v.d.Aalst, M. Dumas, and A. ter Hofstede. Pattern Based Analysis of BPEL4WS. Technical Report FIT-TR-2002-04, QUT, Australia, 2002.

A Business Process-centered Approach for Modeling Enterprise Architectures

Torben Schreiter, Guido Laures

Hasso-Plattner-Institute for IT Systems Engineering,
University of Potsdam, Germany
{torben.schreiter, guido.laures}@hpi.uni-potsdam.de

Abstract: Mastering an Enterprise Architecture is crucial for the entire enterprise to operate most efficiently. However, there is a lack of appropriate means to appropriately visualize and communicate traceability from business processes to other enterprise architectural entities such as e.g. the supporting IT infrastructure. Therefore, systematic in-depth analysis of the entire Enterprise Architecture is barely feasible.

This paper aims to introduce a semi-formal and business process-centered modeling approach for capturing complex Enterprise Architectures based on the well-established modeling techniques *BPMN* (Business Process Modeling Notation) and *UML* (Unified Modeling Language). A meta model formalizing the approach is presented.

It is feasible to apply customizable views tailored to stakeholder-specific concerns providing standardized levels of abstraction to the model in order to improve communication of the big picture of an Enterprise Architecture. A case study illustrates the approach.

1 Introduction

Today's enterprises have to stay competitive with a number of competitors all over the world. Streamlined business processes are a key factor leading to improved competitiveness. However, these processes are embedded in an entire Enterprise Architecture. A proper enactment of the processes is still highly dependent on complex structures in the Enterprise Architecture. This paper presents a business process-centered modeling approach enabling the effective communication of an Enterprise Architecture to different stakeholders.

When speaking of *Enterprise Architectures* (EA), we do not mean architectures of enterprise application systems, which more precisely can be refered to as Enterprise Application Architectures. Indeed, the proper definition of an EA seems useful in order to facilitate a common understanding for the subsequent sections of this paper. The following definition is slightly adapted from [Wi06].

Definition 1. *An* Enterprise Architecture *is the current or future structuring of an organization's processes, information systems, personnel and organizational sub-units, so that they align with the organization's core goals and strategic direction.*

As an enterprise's core competence is based on its EA, the whole enterprise is sentenced to function inefficiently if the enterprise's architecture is inefficient. Of course, enterprise architectures, especially those of large companies, may become extremely complex. Mastering the EA and its complexity is a non-trivial challenge. Therefore, it is our concern to provide a means that can be utilized for the overall improvement of an EA.

The proposed concept for *Enterprise Architecture Modeling* should, consequently, cover various aspects that are important for, firstly, communication of an EA and, secondly, improvement of the EA. The conceptual basis for this is a business process-centered semi-formal modeling approach. The supposed modeling technique enables the visualization of a big picture of the respective EA. Considering the potential complexity of an EA, it seems not feasible to provide this big picture using a single diagram. Instead, a set of stakeholder-specific high level diagrams should be provided.

Modeling an EA, of course, includes IT infrastructural aspects that business tasks rely on. The proposed modeling technique additionally covers organizational aspects that might be necessary for fulfillment of a certain business task. Maintainability and extensibility of the supporting IT infrastructure as well as the identification of business processes that are particularly manpower-intensive are interesting aspects when planning or evaluating an EA. Our modeling technique enables further investigation on issues like these.

This paper is organized as follows: Section 2 will introduce the basic concepts and key objectives that underly the proposed modeling approach. The formal basis of the approach in the form of a MOF-based meta model is presented in the third section. Next, a detailed case study demonstrates a practical example using the modeling approach in the concrete business domain of an insurance company. Finally, a brief survey of related work as well as a conclusion is provided.

2 Conception of the Modeling Approach

2.1 Organizational and IT Infrastructural Aspects

Any modeling technique is based on a number of modeling elements. These elements are then utilized by modelers in order to describe certain aspects of a system of any kind.

When modeling an EA, two different kinds of aspects might be part of a model. These are either *organizational aspects* or *IT infrastructural aspects* of an EA. Thus, the set of entities that are of interest for an EA modeling approach can, in general, be assigned to one of the two kinds of aspects.

Entities such as *business processes*, *organizational (sub-)units*, *entire companies*, a company's *sites* or *roles* cover different organizational aspects of an EA.

On the other hand, e.g. *enterprise applications*, *web services*, *proprietary interfaces*, *business process engines*, *EAI systems* (Enterprise Application Integration), *adapters*, *application servers* or other kinds of servers cover IT infrastructural aspects of an EA. Note that the provided lists are *not* exhaustive.

2.2 Layering Structure

We would like to introduce a layered approach in order to describe an EA. More precisely, a view on an EA model consists of (at most) five distinct layers.

Role layer. The role layer contains organizational roles in an EA. The roles can be required for a particular business task in order to be completed. Roles of an EA may include *customer*, *head of department*, *accounting clerk* amongst others. Additionally, the roles might e.g. be assigned to a department or other (external) companies.

Presentation layer. All IT infrastructural systems related to presentation of digital content to users performing the tasks belonging to a particular role are visualized in the presentation layer. Examples for elements of this layer include different kinds of *portals*, (proprietary) *rich clients* as well as e.g. e-mail clients.

Business process layer. This layer contains elements like entire *business process models*, *process instances* or *tasks* of a process model. Business process models on this layer provide the definition for enactment by a business process engine.

Service layer. The service layer visualizes the service landscape of an EA. Additionally, systems related to the accessability of the services are part of the service layer. These systems are technically inevitable to support the provision of services. However, they do not implement any business functionality. An *application server* utilized for exposure of *web services* set up during service enablement of legacy systems can be named as an example for this.

Please note that even though the layer is named *service* layer, this does not imply that the IT infrastructure necessarily has to be based on the concept of a service-oriented architecture (SOA) as described in [Bu00]. Thus, the service layer might also contain elements like *proprietary interfaces* of legacy backend systems.

Backend layer. Software systems implementing primary business functionality as well as supporting systems are part of the backend layer. E.g. ERP systems (Enterprise Resource Planning), EAI solutions, and relational database management systems may be part of a particular EA.

The business process layer is the *central layer* since we would like to facilitate traceability to concrete business functionality at any time. This layer is modeled using elements taken from the *Business Process Modeling Notation* (BPMN). The remainder of the layers is based on elements taken from UML component diagrams. Please refer to section 3 for further details.

2.3 Dependency Structure

Model elements, and therefore the layers containing these elements, are interconnected by *dependencies*. More precisely, different entities of an EA might be dependent on other entities of the EA and, therefore, require these for fulfillment of a certain business task.

Please find below a definition, which is valid for dependencies of enterprise architectural entities.

Definition 2. *A* dependency $D = (E_1, E_2)$ *exists between two entities* E_1 *and* E_2 *of an Enterprise Architecture if and only if the functionality of* E_1 *depends on the proper functioning of* E_2.

An exemplary dependency might exist between a business task of a business process and a role, stating the fact that human interaction is necessary to fulfill this particular task.

It is important to notice, that the complex interdependent structure of an EA model provides very detailed information on how entities of the EA are interrelated. Moreover, a key property of dependencies is their *transitivity*.

Theorem 1. *Dependencies are* transitive*. This means that if there exists a dependency* $D_1 = (E_1, E_2)$ *and a dependency* $D_2 = (E_2, E_3)$*, then there is also a dependency* $D_3 = (E_1, E_3)$.

Not only due to modern multi-tier architectures of supporting enterprise IT systems, transitiveness of dependencies is, practically, a very common pattern in almost any EA. If, for instance, a business task is performed by invocation of a composite web service, there is, first of all, a dependency between the task and this particular service (the task is dependent on the service). The composite web service is dependent on a number of other web services. And these are again dependent on some backend systems providing the functionality that is exposed by the web services. The backend systems might again be dependent on other backend systems such as a database management system. Because of the transitivity of dependencies, the original business task invoking a web service is also dependent on *all* the other entities involved in the provisioning of this service.

When elaborating further on the example of web services, independence of interface description and its implementation, often, is an important aspect. Indeed, the provided functionality of the service is what matters, regardless of how it is implemented. Nevertheless, when evaluating and analyzing an EA, we want to be able to trace dependencies to as many supporting entities/systems as possible. Naturally, this is mostly impossible when using web services from *external* providers. But still, we are able to derive a dependency from the external web service to its provider (as an organization) since the provider is responsible for the proper functioning of its IT systems. Consequently, the business task relying on this external service is dependent on the external provider.

2.4 Annotation of Enterprise Architectural Entities

Entities that are part of an EA sometimes have several properties or constraints that might be advantageous to be aware of. In an EA model we would like to be able to capture these properties and constraints appropriately. We propose the annotation of model elements as a means to visualize these aspects. Therefore, annotations enable a more detailed description of the EA.

There can be different types of annotations. Very few annotations are implicit like branch-

ing conditions of sequence flows in a business process model for instance. Some other annotations might have to be defined manually such as e.g. the maximum duration of a business task. And then, it is even feasible that some annotations can, to a certain extent, be automatically derived by supporting tools.

Let us consider an exemplary monitoring framework for a business process engine, which is able to provide statistical information on the enactment of processes. If there is a branch in the business process model, normally, the alternative flows are differentiated by conditions. The conditions are now seen as annotations of the sequence flows (arcs) to the next activity. Additionally, the monitoring framework provides information on the probabilities of each branch alternative as a percentage. This would be the probability for taking this particular alternative based on the preceding enactment history of the process. Of course, this can be visualized as another annotation of the sequence flow.

It is an important characteristic of annotations, that they can be transitively passed on to other annotatable elements. Figure 1 depicts the previously introduced example for branching probabilities. In addition, there is a second, nested branch shown. The alternatives again are annotated with condition and probability. *Task 1a* is additionally annotated with a maximum duration of 120 seconds. This task is now dependent on some other entity. All previous annotations are passed on to the dependency. Consequently, the dependency is only relevant if the branch is chosen and this, again, only occurs in 1% of all cases. Please note that the probabilities have been accumulated since a basic pass-on would not be sufficient. The maximum duration has to be handled with care because the duration might be reasonable if there is a single dependency to e.g. a web service. However, if there are multiple dependencies to various types of entities, it might not be reasonable to pass this annotation to any of the dependencies at all.

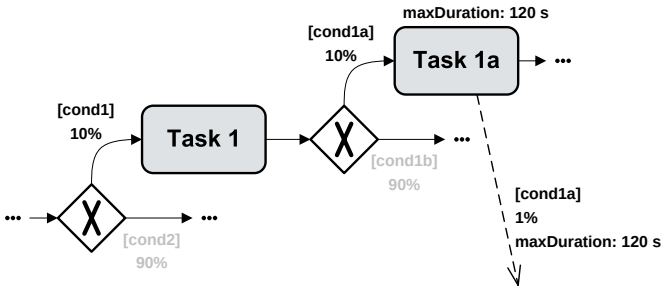


Figure 1: Example for transitivity of annotations.

Additionally, the annotation of *redundancy information* is a very practical example for annotation of dependencies. It is very common that important IT systems such as e.g. application servers are held redundant in order to compensate failure of a machine. If simply denoting a dependency from an entity to each of the redundant machines, we would, according to definition 2, semantically define that the entity is dependent on *each* of these machines. Of course, we would prefer to express that the entity is only dependent on one arbitrary of the machines instead of being dependent on all of them. This is a typical use case for a redundancy annotation.

2.5 Customizable Views

As we declared human communication of the EA to different stakeholders to be a main objective of the proposed modeling approach, it is desirable to satisfy the specific needs of these stakeholders of an EA. We suggest the facility to create customizable views providing precisely defined levels of abstraction. Each view is intended to serve a particular purpose tailored to the needs a stakeholder might have.

One stakeholder towards an EA is possibly a manager. Another stakeholder could be a system administrator. Both have very different attitudes towards the EA. The manager is primarily interested in streamlined business processes whereas the system administrator is concerned with the maintenance of the supporting IT systems. Accordingly, the visualizations should show the aspects that are of interest for the specific person.

Since a number of views can be seen as applicable to different enterprise's architectures, it is viable to provide this generic set as predefined views. Additionally, it should be possible to enable custom view definitions. See section 3 for a formal definition of views as well as section 4 for a view definition example.

A custom view, ideally, only shows the subset of enterprise architectural entities that are of interest for the particular stakeholder it is intended for. The transitivity of dependencies and annotations enables the derivation of implicit dependencies and annotations that exist even though some intermediate entities may not be shown in a particular view. If a business process is e.g. dependent on a number of services and these services are, again, dependent on some backend system, it is possible to derive a dependency from the process to the backend system, even if the whole service layer is undisplayed.

3 The Meta Model

This section aims at formalizing the proposed approach by introducing a meta model. It should be based on the *Meta Object Facility* (MOF) defined by the Object Management Group (OMG) since MOF already provides all concepts needed for definition of a meta model. Please refer to [Ob06a] for further information on the Meta Object Facility. Our modeling approach is based on the well-established modeling techniques *UML* (*Unified Modeling Language*), as specified in [Ob05a] and [Ob05b], and *BPMN* (*Business Process Modeling Notation*), which is specified in [Ob06b].

This paper does not intend to provide a complete specification of the proposed modeling approach. The meta model is rather to be seen as a basic formal definition of how to relate the diagram types UML and BPMN for possible future research on the field of EA modeling.

All layers except for the *Business process layer* are based on UML Component diagrams. As shown in figure 2, the elements of each layer are specializations of the meta class *Component* taken from [Ob05b]. This is advantageous, because we then inherit all properties from the UML meta class component, but still are able to distinguish between different

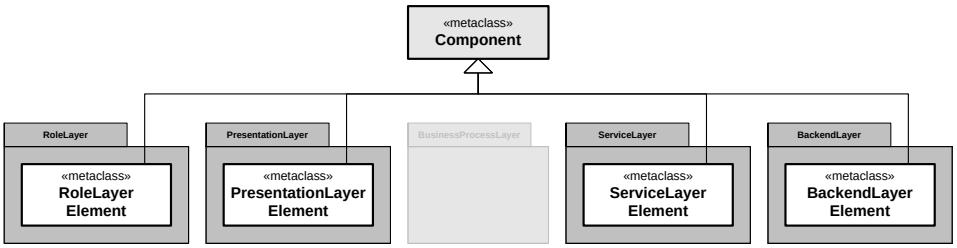


Figure 2: Layer elements based on UML Components.

layer's elements. This, again, is necessary for formal view definitions, as described in section 3.2.

When utilizing UML Components for our purposes, we do not exactly interpret them the way it is stated by the OMG. [Ob05b] describes UML Components as follows:

> [...] In particular, the [Components] package specifies a component as a modular unit with well-defined interfaces that is replaceable within its environment. [...]
>
> [...] The Components package supports the specification of both logical components (e.g., business components, process components) and physical components (e.g., EJB components, CORBA components, COM+ and .NET components, WSDL components, etc.) [...]

Our interpretation of UML Components is more abstract. We propose to model (human) roles and IT systems like application servers as specializations of Components, for instance. In our case, we are more interested in the definition and syntax of Component diagrams rather than in the methodology and intention of UML. Furthermore, we only define very basic elements for these four layers. Currently, further specializations of e.g. the meta class *BackendLayerElement* in form of other meta classes like *Adapter* or *DatabaseManagementSystem* is not intended. However, staying this generic does not pose any difficulty since both the organizational structure as well as the IT landscape of a particular enterprise might consist of a wide variety of elements. Therefore, the elements to be used should not be restricted to a limited set of specific elements but the definition should, instead, be rather abstract in order to be generic enough.

3.1 Annotations and Dependencies

Annotation of elements is accomplished as depicted in figure 3. There is a general, abstract meta class *Annotation* which is always related to at least one *NamedElement*[1]. Furthermore, each *Annotation* may be related to a number of other annotations due to the

[1] *NamedElement* is the meta class taken from UML, we will utilize as base class for all elements of the meta model. Please note that also *Component* is a *NamedElement*.

property of transitivity. This set is derived from the model.

Additionally, there are two specializations of *Annotation*: *DependencyAnnotation* and *BusinessProcessAnnotation*. As stated in section 2.3, the idea of elements is that they can be dependent on other elements. Thus, we relate a *Dependency* (defined in [Ob05b]) to an arbitrary number (including 0) of *DependencyAnnotations*.

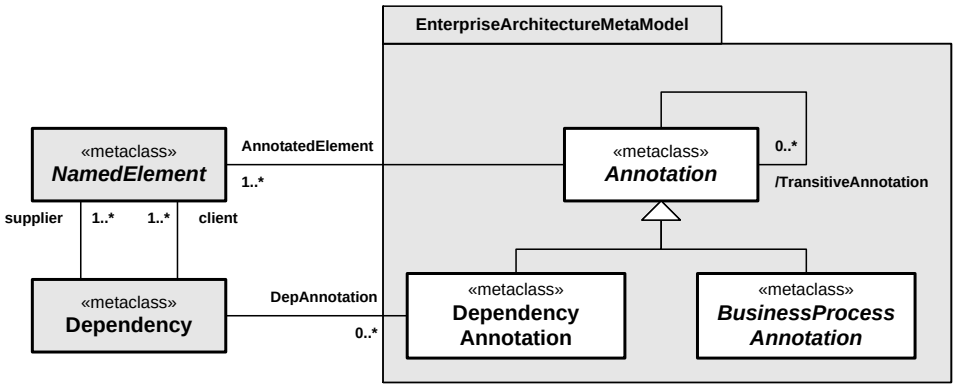


Figure 3: Definition of meta classes related to annotation of elements.

A *Dependency* is defined as an *n:m* relation between *NamedElements* in [Ob05b]. This, at first glance, seems to be not completely aligned with the provided definition in section 2.3. We defined a dependency as binary relation between enterprise architectural entities. However, the *n:m* relation for dependencies defined by [Ob05b] can be seen as an aggregation of multiple dependencies into one modeling element.

Moreover, we introduced a redundancy annotation for dependencies. This can be seen as a special case. Multiple dependencies to different elements annotated as redundant are rather to be seen as a set of dependencies of which only a subset (whose cardinality often equals 1) is necessary for proper functioning.

As we motivated before, the annotation of some elements in the *Business process layer* is desired as well. These annotatable elements are:

- *Activities* (such as tasks and sub-processes)
- *Branching sequence flows* (outgoing arcs of (X)OR gateways)
- *Business processes* (as a whole)

Elements in the Business process layer are based on BPMN. Please note that the current OMG-specification does not provide a formal meta model by itself. Only attributes and types of these attributes are defined in [Ob06b]. However, it makes sense to define at least the most important elements as interrelated meta classes and integrate these into the meta model. Original attribute specifications are, of course, considered.

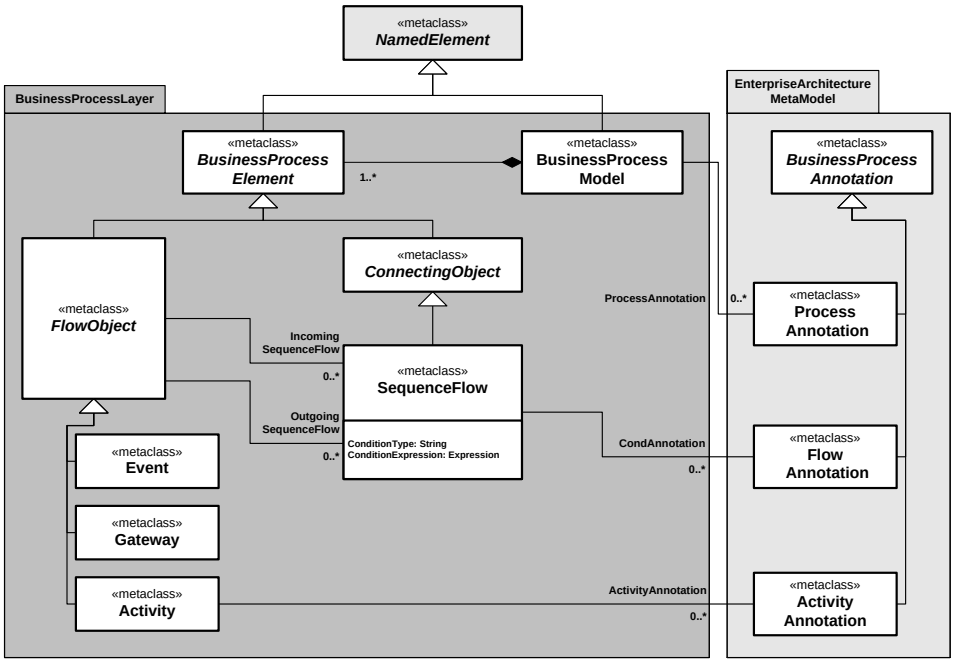


Figure 4: Meta classes for BPMN key elements and relation to annotation.

Figure 4 shows that there are basically two types of elements in the Business process layer. These are *BusinessProcessElements* and *BusinessProcessModels*. A *BusinessProcessModel*, ideally, contains a number of elements. Interdependency of the elements is facilitated due to the specialization of the meta class *NamedElement*. As defined by [Ob05b], the meta class *Dependency* relates any *NamedElements*.

The abstract meta class *BusinessProcessElement* is specialized into *FlowObjects* and *ConnectingObjects*. These meta classes are directly appendant to the corresponding concepts of BPMN. A *FlowObject* can be an *Event*, *Gateway*, or an *Activity*. A *ConnectingObject* can e.g. be a *SequenceFlow*.

Furthermore, there are three different *BusinessProcessAnnotations* and their respective associations to elements of the Business process layer shown. An *Activity* might be annotated with an *ActivityAnnotation*. This is similar for *BusinessProcessModel* and *ProcessAnnotation*. But then it is slightly different in case of the *FlowAnnotation*. [Ob06b] defines the attributes *ConditionType* and *ConditionExpression* enabling to distinguish whether a *SequenceFlow* contains a condition or not. Since simple sequence flows do not carry an additional meaning and it is, therefore, not reasonable to annotate these, we have to restrain the use of a *FlowAnnotation* to those *SequenceFlows* carrying anything but the value *"None"* for the attribute *ConditionType*.

3.2 View Definition

In section 2.5 we described the use of customizable views on the EA model. Now, the mechanisms for defining these custom views are introduced.

The definition of views is based on the facility (provided by UML) to define Profiles and apply these on diagrams [Ob05b]. A *View*, thus, is a specialization of the meta class *Profile* as depicted in figure 5.

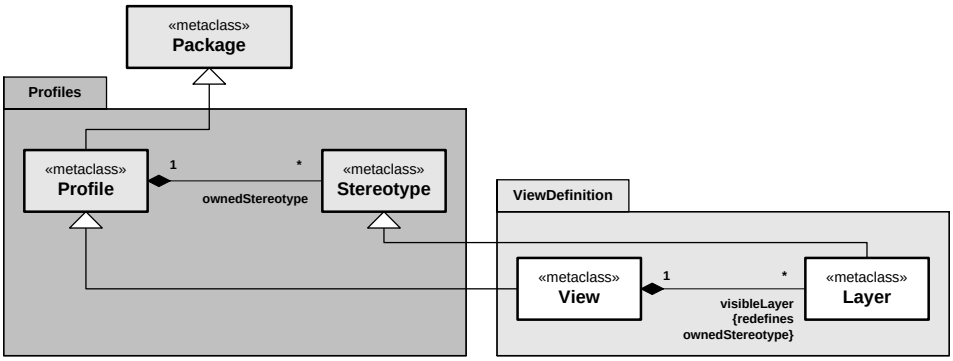


Figure 5: Definition of stakeholder-specific views.

Originally, a UML Profile contains a number of *Stereotypes*. These Stereotypes can then be applied to meta classes. Similarly, a *View* contains *Layers* in our case. Usually, there should be five or less distinct layers used for the definition of a view[2]. Each of which corresponds to the layers presented in section 2.2. A custom view then defines the elements that are to be shown by assigning a meta class to a particular layer. This is accomplished by using the UML concept *Extension* as defined in [Ob05b].

Finally, a view is applied to the EA model. As a result, only the elements based on the restricted set of meta classes is displayed. Please refer to section 4 for an example of view definition.

4 Case Study

This section introduces a case study in order to illustrate the presented EA modeling approach. Similarly to [PT06], this case study will be situated in the business domain of a fictive insurance company. The resulting EA model, thus, describes (a part of) the enterprise architecture of this company. Since our approach is business process-centered, we necessarily need information on the business processes that are part of day-to-day business of the insurance company. For simplicity reasons we assume, that the business processes

[2]Please note that we do not intend to restrain the number of layers to be defined in a view. This would, theoretically, facilitate the definition of multiple layers of the same kind or even other layers. However, we recommend to only define one layer of the same kind per view.

have already been defined and optimized. We furthermore assume that the business processes' activities are supposed to be (partially) implemented by web services. Invocation of the services should be realized using a BPEL engine.

As part of a service enablement project the company needs to define its EA model in subsequent steps to be able to identify the IT landscape's pain points as well as to track the project's progress. The first step is to define the concrete dependency structure for each of the web services to be developed. This means that for each web service, it should be described, which IT systems provide its functionality. The resulting dependency structure is the basis for the EA model. Other parts of the model can be derived from the BPEL processes, since these already define associations from the business activities to the implementing web services. Some other dependencies have to be defined manually.

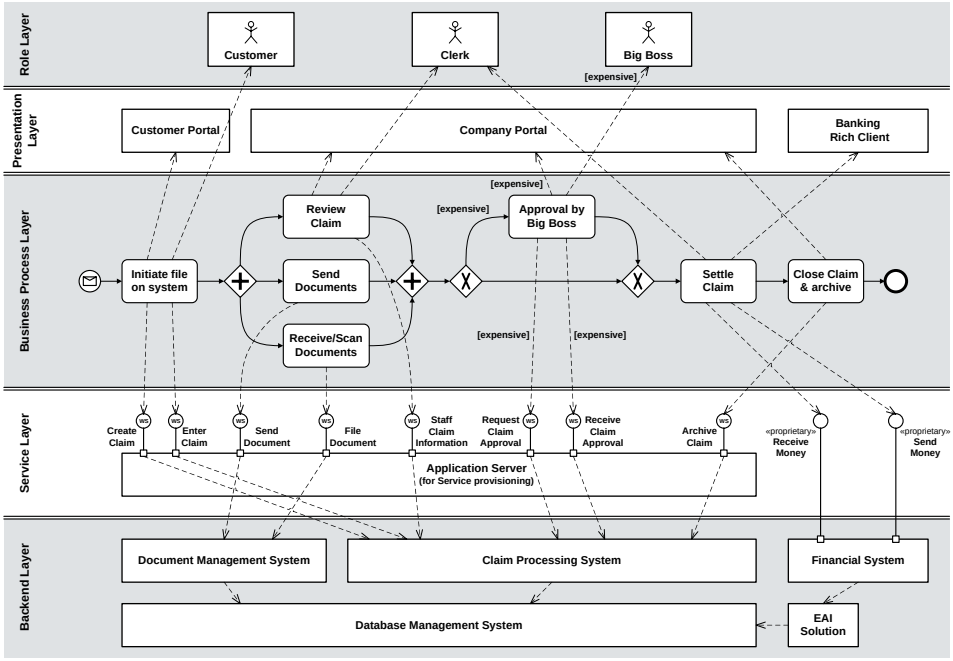


Figure 6: View on all layers of the EA model for the business process *ProcessInsuranceClaim*.

After the EA model is constructed using our modeling approach, it is possible to define views on the model. Figure 6 shows the complete dependency structure for the simple business process *ProcessInsuranceClaim*. This view includes all five layers. Starting from left to right in the business process layer, the customer first has to initiate the claims processing by informing the insurance company of the claim. In this example, the customer enters the claim details through the Internet using a customer portal. The web services *CreateClaim* and *EnterClaim* are invoked as a result of the customer's initiation of claims processing. A dedicated application server is responsible for the exposure of these web

services. It is not responsible for the services' implementation[3]. The services depend on the backend systems *Claim Processing System* and a *Database Management System*. After the initiation, a clerk reviews the claim. During the process of reviewing a claim, several documents have to be sent and others are received. After the claim was reviewed, it depends on the arising expenses whether the clerk's superior has to approve the settling of the claim or not. Consequently, the dependencies to the web services as well as to the role are annotated with the condition expression of the branching sequence flow in the business process. When settling the claim, the clerk has to use a rich client for the bank transfers, since the company portal does not support this functionality. This rich client uses proprietary interfaces of a financial backend system, which is not (yet) service enabled. Finally, the claim is closed and archived.

This example of a simple business process illustrates very well, how complex an EA can become when considering several dozens of more complex business processes and in other conceivable cases an even more heterogeneous IT landscape.

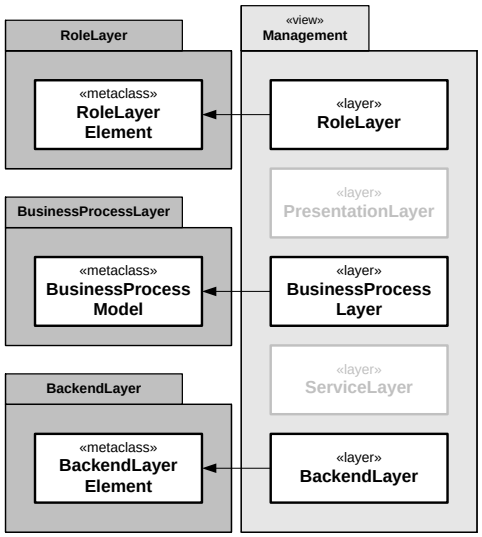


Figure 7: Definition of the Management view.

A view showing all layers as depicted in figure 6 is very detailed and contains (technical) aspects such as particular web services, which are not of primary interest for the insurance company's management. Thus, we define a custom view in order to meet the specific needs of the managers. Figure 7 shows this view definition. Presentation layer as well as the service layer should not be displayed at all. The business process layer should contain business process models instead of a single process model's details.

This results in a high level view on the EA model as depicted in figure 8. It shows two business process models in the business process layer. The first one is the same process as described above. It is obvious that the dependencies of the respective business tasks

[3]Please see section 2.2 for an explanation.

were aggregated into dependencies of the business process model. Additionally, the dependencies to backend systems were derived from the ones that exist between elements in the service layer and the backend systems.

The second business process describes the regularly performed billing of a customer, which is fully automated. Thus, the process model is not dependent on any role. On the other hand, there are dependencies to a billing system, which again is dependent on different other systems.

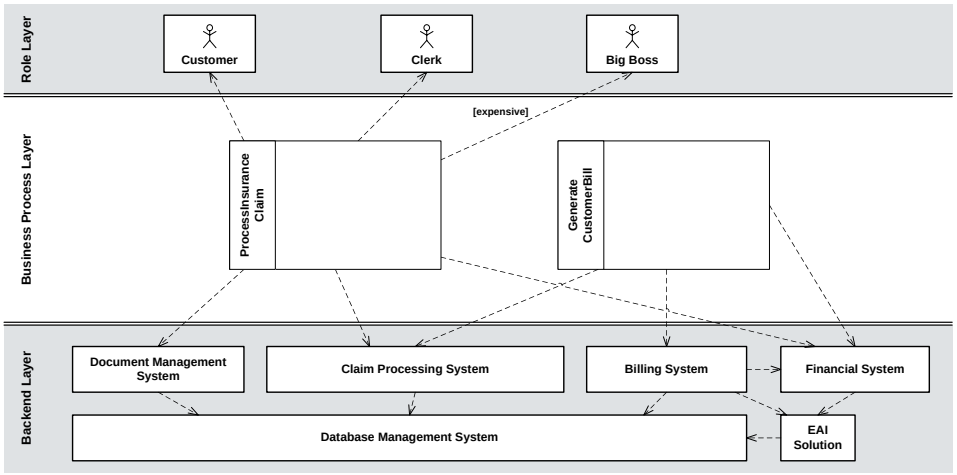


Figure 8: Management view applied on the EA model.

Diagrams like these help to communicate a big picture of the EA to different stakeholders. In this particular case, the management is able to establish an understanding of how business processes are related to organizational roles and the basic IT infrastructure. Apart from monitoring the service enabling project it is also conceivable to e.g. track how many processes rely on the recently purchased document management system. In combination with information on the status of employees (role instances) based on e.g. sick reports and vacation, an identification of bottlenecks is possible.

Similarly to the management view, it is possible to define a maintenance view concentrating the aspects that are of interest for the maintenance staff.

5 Related Work

The approach presented in this paper is the logical continuation of the IT landscape specification formalized by Breest [Br05]. It bases on well settled standards like the Unified Modeling Language [Ra97] and the Business Process Modeling Notation [Ob06b]. We combined the meta models of these techniques to form a new meta model for the entire EA. There have already been efforts spent to define comparable models.

The Common Information Model (CIM) [Dm05] provides a common definition of management information for systems, networks, applications and services. However, it is not possible to apply CIM to functional elements like business processes. It focuses more on the structural elements which are settled in the lower layers of our approach. Our means for dependencies is slightly aligned with the CIM definition.

Another approach to define a formal notation for all kinds of elements of an IT infrastructure is the Reference Model for Open Distributed Processing (RM-ODP) [Is95]. Much like the approach presented in this paper the RM-ODP defines viewpoints on a model in order to provide stakeholder-specific views. However, this model focuses much on technical issues of distributed information processing (e.g. streaming) and does not deal with enterprise entities from our higher layers (e.g. roles, business processes).

The Unified Modeling Language concentrates on the analysis and design of object-oriented applications. The coarse-grained entities of an EA are poorly reflected in the core concepts of this language. The dynamic and functional parts of the UML are rarely used to specify business processes. They are best suited for specifications of internal application flow. However, the concept of UML profiles is very useful for the definition of different views and the UML meta model provides the extension capabilities to define new concepts. Thus, we decided to use UML profiles for view definition and to derive our new modeling concepts from the UML meta model.

The ARIS tool set [Sc98] is widely spread in today's industries to define high level business processes. It uses the Event-driven Process Chain (EPC) [KNS92] notation to specify business processes of enterprises. However, it lacks a formalization for the business process reflection in the implementing IT infrastructure. Thus, it does not provide the traceability we need for our approach.

The Web Service Architecture [W304] describes a lot of concepts surrounding the entire web service area. These contain also entities like processes, tasks, agents and so on. However, it is less formal than the approach we presented. The dependencies described within this architecture aim at a basic understanding of a typical web service scenario rather than on a formal specification that could support use cases like those mentioned before.

The ArchiMate project [Jo03] elaborated an EA specification language. Our approach is very much aligned to that from this project, however, we focus more on the transitivity of dependencies between the architectural entities as well as on the concept of annotation.

Although the above related work results show that there is a complete language coverage of separate architectural concepts, there is still a lack of integration between them. We, therefore, focused on an integration and reuse of the best-of-breed of these languages.

6 Conclusion

We presented a business process-centered modeling approach aiming at the description of complex dependency structures within an Enterprise Architecture. The dependency structure is implicitly existent in any EA. Due to efficient visualization of the dependencies and

annotation of these dependencies with non-functional information, it is, for the first time, possible to *explicitly* visualize and communicate these complex structures. The approach is based on the well-established modeling techniques UML and BPMN. A MOF-based meta model was provided, formalizing the approach.

Furthermore, we provided a facility to define customized views on the EA model tailored to the specific needs of different stakeholders. This enables the effective communication of complex EAs in form of a big picture of the EA covering the aspects of interest for each particular stakeholder.

Finally, the approach was illustrated in a case study. A practical use of the approach was depicted. The added value of the approach during and after a service enablement project features improvement of communication and analysis as well as optimization of the EA.

However, there is still a number of aspects that might be subject of further research. Various (semi-)automated auditing techniques for an EA such as *impact analysis*, *availability analysis*, and *performance analysis* based on metrics and heuristics are conceivable. All auditing is enabled due to traceability of dependencies between enterprise architectural entities and their annotation. Depending on the kind of analysis, automated report generation might be desirable. In other cases, it might be interesting to interactively simulate certain scenarios in order to investigate issues that, otherwise, would only appear when actively modifying the EA. So, it is, to a certain extent, possible to investigate issues that are mostly inherent with major changes to an EA even before actually changing the EA. Concrete heuristics and metrics are to be identified in order to facilitate auditing of EAs.

Next, our approach is based on the assumption that it can be seamlessly integrated into a business process engine enacting the business processes. This integration might prove to be a complex task. Then, for reasonable and extensive annotation of elements with non-functional information it is, possibly, necessary to access different frameworks such as e.g. a monitoring framework providing statistical information on process enactment.

Generally, the approach should be applied in a real company in a productive environment in order to prove its added value and to, eventually, improve or redesign details. It might, for instance, turn out that a hierarchical structuring of organizational entities (such as companies, departments, roles) is advantageous in certain situations.

References

[Br05] Breest, M.: Specifying Service Landscapes. Seminar Reader of the Hasso-Plattner-Institute, 2005
http://bpt.hpi.uni-potsdam.de/twiki/pub/Public/SeminarPublications/ReaderBPM2.pdf

[Sc05] Schubert, H.: Entwicklung eines QoS-Frameworks für Service-orientierte Architekturen. Masters Thesis. Hasso-Plattner-Institute for Software Systems Engineering, SAP Systems Integration AG, 2005

[Bu00] Burbeck, S.: The Tao of e-business Services. Emerging Technologies, IBM Software Group, 2000

[PT06] Pulier, E., Taylor, H.: Understanding Enterprise SOA. Manning Publications Co., 2006

[TGK06] Tabeling, P., Gröne, B., Knöpfel, A.: Fundamental Modeling Concepts - Effective Communication of IT Systems. John Wiley & Sons, Ltd, 2006

[Ob05a] Object Management Group: UML Infrastructure Specification, v2.0, 2005

[Ob05b] Object Management Group: UML Superstructure Specification, v2.0, 2005

[Ob06a] Object Management Group: Meta Object Facility (MOF) Core, v2.0, 2006

[Ob06b] Object Management Group: Business Process Modeling Notation (BPMN) Specification, 2006

[Wi06] Wikipedia: Enterprise architecture, July 2006
http://en.wikipedia.org/w/index.php?title=Enterprise_architecture&oldid=64463956

[Ra97] Rational Software (editor): UML Notation Guide 1.1, Unified Modeling Language Version 1.1. Santa Clara (USA), 1997

[Dm05] DMTF: CIM: Common Information Model Schema Version 2.11, December 2005
http://www.dmtf.org/standards/cim/cim_schema_v211

[Is95] ISO/IEC: ITU-T X.901 ISO/IEC 10746-1 Open Distributed Processing Reference Model Part 1. Draft International Standard (DIS) output from the editing meeting in Helsinki (Finland), May 1995

[Sc98] August-Wilhelm Scheer: Aris-Business Process Frameworks. Berlin; New York: Springer, 1998

[KNS92] G. Keller and M. Nüttgens and A.-W. Scheer: Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Saarbrücken, 1992

[W304] W3C: WSA: Web Services Architecture, February 2004
http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[Jo03] Henk Jonkers, Buuren Buuren, Farhad Arbab, Frank de Boer, Marcello Bonsangue, Hans Bosma, Hugo ter Doest, Luuk Groenewegen, Juan Guillen Scholten, Stijn Hoppenbrouwers, Maria-Eugenia Iacob, Wil Janssen, Marc Lankhorst, Diederik van Leeuwen, Erik Proper, Andries Stam, Leon van der Torre, Gert Veldhuijzen van Zanten: Towards a Language for Coherent Enterprise Architecture Descriptions. Seventh International Enterprise Distributed Object Computing Conference (EDOC), p. 28., 2003

Flexibilitätsanalyse service-orientierter Architekturen zur Realisierung von Geschäftsprozessen

Guido Laures

Hasso-Plattner-Institut für Softwaresystemtechnik GmbH
Prof.-Dr.-Helmert-Strasse 2-3, 14482 Potsdam, Germany
guido.laures@hpi.uni-potsdam.de

Abstract: Die Effizienz eines Geschäftsprozesses hängt zum großen Teil von dessen Automatisierungsgrad ab. Aus diesem Grund gehört Prozessautomation zu einer der wichtigsten Erfolgsfaktoren innerhalb einer IT-Organisation. Durch die Einführung einer unternehmensweiten, service-orientierten Architektur versprechen sich Unternehmen eine höhere Flexibilität.

Beim Design und der Implementierung eines Service können jedoch Fehler begangen werden, die dazu führen, dass die Investitionen in die Einführung von Services nicht in gesteigerter Prozesseffizienz münden und damit das ursprüngliche Ziel verfehlen. In dieser Arbeit werden Muster innerhalb einer service-orientierten Architektur auf ihre Flexibilität hin untersucht und bewertet. Die dabei entstehende Mustersammlung unterstützt IT-Architekten beim Service- und Prozessdesign und hilft damit a priori unflexible IT-Architekturen zu vermeiden.

1 Einführung

Die Optimierung der Flexibilität der IT-seitigen Realisierung von Geschäftsprozessen ist oft eines der Hauptziele bei der Einführung einer unternehmensweiten service-orientierten Architektur (SOA) [PvdH06]. Die Vorstellungen davon, was in diesem Zusammenhang genau unter Flexibilität zu verstehen ist, divergieren jedoch signifikant. Was fehlt, ist eine Kategorisierung der architekturellen Konzepte, die innerhalb einer SOA üblicherweise verwendet werden, und deren Bewertung nach formalen Flexibilitätskriterien. Die daraus ableitbaren Erkenntnisse können zukünftig vermeiden, dass schon in frühen Phasen eines SOA-Projektes Fehler begangen werden, die die angestrebten Projektziele gefährden.

Diese Arbeit lenkt den Blick beim Entwurf und der Realisierung einer Unternehmens-SOA auf die Flexibilität der Implementierung der umzusetzenden Geschäftsprozesse. Hierzu wird nach Abgrenzung des Begriffs der Unternehmensarchitektur eine Definition sowie eine Berechnungsvorschrift für Flexibilität gegeben. Darauf bezugnehmend werden Architekturmuster erarbeitet, auf Flexibilität hin untersucht und anhand einer Fallstudie erläutert. Die erarbeiteten Muster werden daraufhin zusammenfassend und vergleichend gegenübergestellt und bewertet bevor themenverwandte Arbeiten gegen den in diesem Papier vorgestellten Ansatz diskutiert werden. Schließlich wird ein Ausblick gegeben, wie die in dieser Arbeit vorgestellten Erkenntnisse weitergeführt und ausgeweitet werden können.

2 Grundlagen

2.1 Unternehmensarchitekturen

Unternehmensarchitekturen werden von Zachmann [Zac97] als das *Thema des Jahrundert* bezeichnet. Der Begriff *Architektur* wird hier als die Menge der Entwurfsartefakte und beschreibender Repräsentationen bezeichnet, die notwendig sind, um ein Objekt so zu beschreiben, dass es mit gleicher Qualität reproduziert und über seine Lebensdauer gewartet werden kann. Ein Unternehmen (Enterprise) ist nach Wegmann [Weg03] eine Organisation von Ressourcen, die Prozessen unterliegt. Beispiele für solche Ressourcen sind Menschen, Computer, Maschinen oder auch Gebäude. Eine Unternehmensarchitektur (Enterprise Architecture) ist eine reproduzierbare und wartbare Beschreibung eben solcher Ressourcen.

Da Unternehmensarchitekturen der Untersuchungsgegenstand dieser Arbeit sind, ist es wichtig, sie von Architekturen zur Implementierung von Unternehmensanwendungen abzugrenzen. Die ebenfalls oft als Enterprise Frameworks bezeichneten Technologien wie J2EE oder .NET bilden nach unserem Verständnis keine Unternehmensarchitekturen. Gleiches gilt für Architekturen von Unternehmensanwendungen wie sie beispielsweise Fowler [Fow02] behandelt werden. Diese beschreiben applikationsinterne Architekturkonzepte, die im Gegensatz zu applikationsübergreifenden für eine Unternehmensarchitektur nicht relevant sind. Der folgende Abriss der historischen Entwicklung von Unternehmensarchitekturen soll dieses Verständnis weiter untermauern.

Service-orientierten Architekturen sollen in IT-Projekten den Fokus weg von technischen hin zu prozessrelevanten Aspekten der Unternehmensarchitektur leiten. Technische Aspekte bestimmen jedoch in SOA Projekten oft den Planungs- und Projektalltag und priorisieren technische Integration höher als die Transparenz von Unternehmensprozessen in den IT-Systemen der Unternehmensarchitektur. Auch bei der Verwendung von Services zur Abbildung von Geschäftsfunktionalitäten besteht die Gefahr, unflexible Gesamtarchitekturen zu erhalten. Es geht nicht nur um die korrekte, technologische Bereitstellung von Services, sondern auch um deren strukturierte Einbindung und Verwendung innerhalb der Unternehmensarchitektur.

2.2 Flexibilität

Die Abbildung von Geschäftsprozessen in den Strukturen der sie realisierenden IT-Landschaften ist sowohl modellierungsseitig als auch in Bezug auf Wartung eine der größten Herausforderungen der IT-Organisation eines Unternehmens [DS90, DHL01, LRS01]. Formale Kriterien zur Bewertung der Güte einer Reflektion von Geschäftsprozessen innerhalb einer IT-Architektur sind jedoch schwer zu finden. Das Ziel der Flexibilisierung wird zwar allgemeinhin als erstrebenswert hingestellt [PvdH06, NJFM96, Zac99] jedoch wird dessen Erreichungsgrad selten formal geprüft.

In [Wik06] wird Flexibilität als die Fähigkeit bezeichnet, sich an verschiedene Begebenheiten anzupassen. Auf Unternehmensarchitekturen und Unternehmensprozesse bezogen

Abbildung 1: Die Beispielprozesse p_1 und p_2 bestehen lediglich aus einer simplen Sequenz von zwei Tasks

bedeutet Flexibilität also die Fähigkeit einer IT-Landschaft, auf ändernde Geschäftsprozesse hin angepasst zu werden.

Die Flexibilität eines Systems bzgl. der Umsetzung von neuen oder modifizierten Geschäftsprozessen verhält sich anti-proportional zu den Aufwänden, die für diese Umsetzung betrieben werden müssen. Zudem sollte zur Bestimmung der Flexibilität auch die resultierende Wartbarkeit eines Systems nach Umsetzung notwendiger Modifikationen berücksichtig werden. Hierzu gilt die Regel, dass die Flexibilität als schlechter zu bewerten ist, sofern die Wartbarkeit des Systems nach Umsetzung einer Änderung in der Prozesslandschaft gesunken ist.

Sei S_P eine Menge von Systemen, die die Prozessmenge P realisiert. Sei P' eine weitere Prozessmenge und $T_{S_P,S_{P'}}$ die Menge der Transformationsschritte, die S_P durchlaufen muss, um es in $S_{P'}$ zu überführen. Seien weiterhin $E_{T_{S_P,S_{P'}}}$ die Summe der Aufwände der Elemente von $T_{S_P,S_{P'}}$ und M_S die Wartbarkeit einer Systemmenge S. Dann bezeichnet

$$F_{S_P,P'} = \frac{M_{S_{P'}}}{1 + E_{T_{S_P,S_{P'}}}}$$

die Flexibilität von S_P bzgl. einer Überführung in $S_{P'}$.

Hierbei wird davon ausgegangen, dass $E_{T_{S_P,S_{P'}}} \geq 0$ gilt und damit

$$T_{S_P,S_{P'}} = \{\} \Rightarrow E_{T_{S_P,S_{P'}}} = 0 \Rightarrow F_{S_P,P'} = M_{S_{P'}}$$

sicherstellt. Zudem wird festgelegt, dass $0 < M \leq 1$ und damit auch $0 < F \leq 1$ gilt.

2.3 Fallstudie

Das folgende Fallbeispiel wird im nächsten Abschnitt dazu verwendet, die unterschiedlichen Flexibilitätsstufen, die innerhalb service-orientierter Architekturen herrschen können, zu diskutieren. Das Beispiel setzt voraus, dass Tasks innerhalb von Geschäftsprozessen grundsätzlich auf Services innerhalb einer Unternehmens-SOA abgebildet werden können. Diese Vorraussetzung ist zwar eher unrealistisch, da sich derzeit die meisten Unternehmen erst auf dem Weg hin zu einer solchen Architektur befinden [STM+04]. Es zeigt sich jedoch anschaulich, dass selbst in Unternehmen, in denen diese Vorraussetzung gilt, die verwendeten Muster zur Service-Realisierung über den Flexibilitätsgrad entscheidet [Hef04].

Im untersuchten Unternehmen sei die Prozessmenge P bestehend aus zwei sehr einfachen Geschäftsprozessen p_1 und p_2 gegeben (siehe Abbildung 1). Prozesse sind in dieser Arbeit

Abbildung 2: Der neu eingeführte Prozess p_3 komponiert Tasks der Prozesse p_1 und p_2

in BPMN [OMG06] spezifiziert. In Zukunft sollen die Systeme des Unternehmens jedoch die Prozessgruppe $P' = P \cup \{p_3\}$ unterstützt werden. Der neu eingeführte Prozess p_3 beinhaltet Tasks aus beiden Prozessen aus P. Dies ist ein übliches Szenario, bei dem existierende Tasks mit neuen zu einem neuen Prozess komponiert werden. Der neu eingeführte Prozess ist in Abbildung 2 dargestellt.

Im folgenden werden Realisierungsmuster der Prozessgruppe P daraufhin untersucht, wie flexibel sie in Bezug auf eine Realisierung der Prozessgruppe P' sind.

2.4 Notation

Zur grafischen Darstellung der systemseitigen Realisierung eines Prozesses wird die Notation aus Tabelle 1 verwendet. Das dabei verwendete Konzept für systemseitige Serviceimplementierungen ist angelehnt an den Agentenbegriff aus der Web Service Architecture der W3C [W3C04].

3 Muster

In diesem Abschnitt werden eine Reihe von Mustern vorgestellt, die innerhalb einer Unternehmensarchitektur vorliegen können. Wir beschränken uns bei der Identifikation von Mustern jedoch nicht auf die technologische Struktur der Systeme und deren Abhängigkeiten. Vielmehr stellen wir diese in Relation zu den Geschäftsprozessen, die durch sie implizit oder explizit realisiert werden. Nach einer allgemeinen Beschreibung jedes Musters, wenden wir es auf die Fallstudie an, um es in den Kontext von SOA zu stellen. Abschließend wird die Flexibilität jedes Musters untersucht und anhand der oben gegebenen Formel errechnet.

3.1 Direkter Aufruf

Das erste untersuchte Muster ist wohl aufgrund seiner Trivialität auch das am häufigsten in einer Unternehmensarchitektur auftretende. Beim Muster *Direkter Aufruf* wird eine Transition zwischen zwei Tasks t_1 und t_2 innerhalb eines Geschäftsprozess direkt vom realisierenden System a_1 des Tasks t_1 implementiert (vgl. Abbildung 3). Das System a_1 ruft also das System a_2 direkt auf, um somit implizit den Prozess zu implementieren.

	System	Innerhalb einer Unternehmens-SOA ist dies meist eine Geschäftsanwendung.
	Service	System, das den Service-Agenten implementiert. Für gewöhnlich ist dies nicht gleichzusetzen mit dem System, das die eigentliche Service-Funktionalität implementiert.
	Service-Realisierung	Service-Aufrufe von s_1 werden von a_1 realisiert. a_1 implementiert also die Geschäftsfunktionalität, die zur Realisierung der von s_1 bereitgestellten Service-Schnittstelle benötigt wird.
	Trigger	Nicht näher spezifizierter interner oder externer Trigger. Dies kann beispielsweise ein anderes System oder auch ein interner Zeitgeber sein.
	semantischer Bezug	Semantischer Bezug von Prozesselementen und deren systemseitiger Realisierung.
	systemseitiger Prozessfluss	Systemseitiger Fluss des realisierten Geschäftsprozesses. Die Nummer bestimmt die Reihenfolge innerhalb des Fluss.
	systemseitige Prozessfluss-Beendigung	Symbolisiert die Stelle, an der ein Prozessfluss durch die Rücklieferung eines Ergbnis oder durch die Ausführung einer Aktion beendet wird.

Tabelle 1: Verwendete Notation

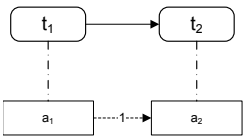


Abbildung 3: Direkter Aufruf

3.1.1 Fallstudie

Wendet man dieses Muster auf obige Fallstudie an, so ergibt sich für die Prozessgruppe P innerhalb einer service-orientierten Architektur die Realisierungen der Abbildung 4. Da wir in unserer Fallstudie von einer service-orientierten Architektur ausgehen, sind die aufgerufenen Systeme jetzt die zu den Tasks des Geschäftsprozesses semantisch äquivalenten Services. Würde im Prozess p_1 a_1 nicht s_2 sondern stattdessen direkt a_2 aufrufen, so könnte man nicht mehr von Service-Orientierung sprechen.

Es ist an dieser Stelle erwähnenswert, dass a_1 zur Realisierung der dargestellten Prozesse zur Laufzeit über Prozesskontextinformationen, also Informationen darüber, welcher Prozess gerade durchlaufen wird, verfügen muss. Dies ist deshalb der Fall, da eine Laufzeitentscheidung getroffen werden muss, ob bei Aufruf des Service s_1 der Service s_2 oder

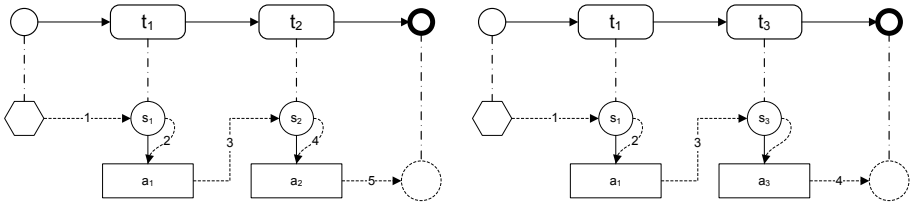


Abbildung 4: *Direkter Aufruf* innerhalb einer SOA

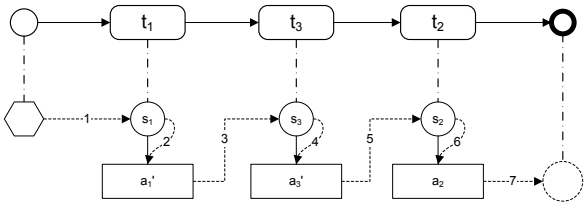


Abbildung 5: Realisierung von p_3 mit Muster *Direkter Aufruf*

der Service s_3 aufgerufen werden muss. Diese Information muss beim Service-Aufruf von s_1 mitgegeben werden. Eine solche implizite Abhängigkeit von Prozesskontextinformationen wirkt sich offensichtlich negativ auf die Wartbarkeit eines Systems aus.

3.1.2 Flexibilitätsbewertung

Um die Flexibilität dieses Musters zu bewerten, betrachten wir nun das Verhalten der systemseitigen Realisierung bei Einführung des neuen Prozesses p_3. Abbildung 5 zeigt, dass a_3 modifiziert werden muss, um den neuen Prozess realisieren zu können. Dies ist dadurch begründet, dass a_3 bisher nach Aufruf durch s_3 den Prozessfluss beendete (p_2). In p_3 jedoch muss a_3 s_2 aufrufen, um den Prozess korrekt abzubilden. Dies wird wieder durch Kontextinformationen im Aufruf von s_3 realisiert. Unter Umständen muss auch a_1 modifiziert werden. Dies wird dann notwendig, wenn die von a_3 benötigten Kontextinformationen nicht mittels der bestehenden Implementierung von a_1 durchgeleitet werden können.

Um den Wert der Flexibilität dieses Musters zu bestimmen und mit den nachfolgenden Mustern zu vergleichen werden nun für bestimmte Größen Referenzwerte festgelegt. Die Wartbarkeit des Musters *Direkter Aufruf* (DA) bezeichnen wir mit M_{DA}, den Aufwand zur Überführung von S_P nach $S_{P'}$ als E_{DA} und erhalten gemäß obiger Formel als Wert für die Flexibilität dieses Musters

$$F_{DA} = \frac{M_{DA}}{1 + E_{DA}}.$$

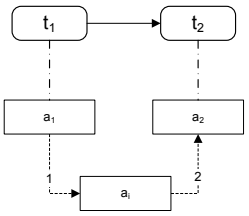


Abbildung 6: Indirekter Aufruf

3.2 Indirekter Aufruf

Das Muster *Indirekter Aufruf* fügt dem Muster *Direkter Aufruf* eine Indirektion hinzu. Zwischen dem System a_1, das den Task t_1 implementiert, und a_2, das den darauffolgenden Task t_2 implementiert liegt hier ein zusätzliches System a_i, das von a_1 aufgerufen wird und den Aufruf nach a_2 weiterleitet (vgl. Abbildung 6). Das weiterleitende System kann demnach auch als Proxy [GHJV94] bezeichnet werden.

3.2.1 Fallstudie

Es kommt oft vor, dass a_i Datentransformationen durchführt, um Inkompatibilitäten von Datenformaten zu überbrücken. Es ist somit ein typisches Szenraio, wie man es in Architekturen antrifft, die EAI Produkte verwenden. Die Funktionalität von a_i ist eine typische Eigenschaft eines solchen EAI Produktes wie beispielsweise eines Integration Broker. Abbildung 7 veranschaulicht die Realisierung von P' mittels des Musters *Indirekter Aufruf*. Auch hier wird davon ausgegangen, dass a_i die vorliegende SOA aus der Fallstudie verwendet, um indirekte Aufrufe zu realisieren. Dies bietet sich schon deshalb an, da die meisten EAI Produkte bereits in der Lage sind, gängige Service-Schnittstellen und Protokolle (z.B. SOAP) zu bedienen. Wenn a_i direkt auf die Geschäftsanwendungen a_2 und a_3 verweisen würde, würde man diese Eigenschaft ignorieren und Mehraufwand bei der Realisierung von Integrationsszenarien in Kauf nehmen.

3.2.2 Flexibilitätsbewertung

Durch die Explizitmachung der Kommunikation zwischen den Geschäftsanwendungen innerhalb von a_i wird auch die Realisierung der Prozesstransformationen sichtbarer und damit leichter zu warten. Jedoch wird die Systemlandschaft durch die Hinzunahme des zusätzlichen Systems a_i komplexer und deren Wartung erfordert zusätzliches Produktwissen. Somit kann man die Wartbarkeit M_{IA} dieses Musters mit der des Musters *Direkter Aufruf* M_{DA} gleichsetzen ($M_{DA} = M_{IA}$). Es ist zudem davon auszugehen, dass a_i darauf ausgelegt ist, exakt solche Änderungen zu unterstützen, die notwendig sind, um P' zu unterstützen, da dies eine Kernfunktionalität von Integration Brokern darstellt. Der Aufwand E_{IA} zur Realisierung von P' in diesem Muster ist somit geringer als E_{DA}. Hiermit ergibt

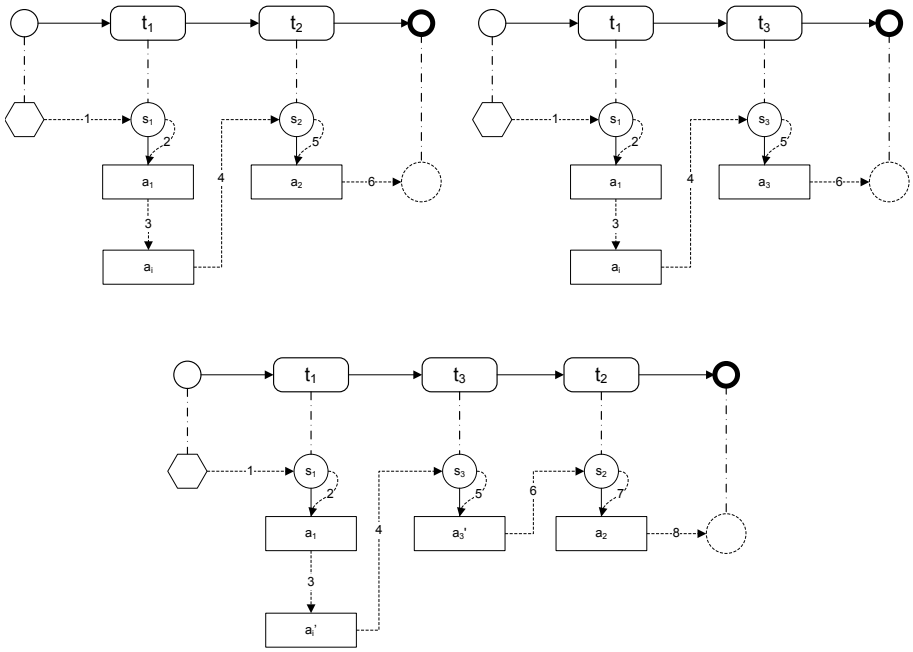


Abbildung 7: Realisierung von P' mit Muster *Indirekter Aufruf*

sich für die Flexibilität F_{IA} dieses Musters

$$F_{IA} - F_{DA} = \frac{M_{DA}}{1 - E_{IA}} - \frac{M_{DA}}{1 - E_{DA}} > 0,$$

da $E_{IA} < E_{DA}$, womit gezeigt ist, dass dieses Muster eine höhere Flexibilität bietet als das Muster *Direkter Aufruf*.

3.3 Vermittelter Aufruf

Beim Muster *Vermittelter Aufruf* konsultiert das System a_1 ein weiteres System a_{ref} bevor es selbst den Aufruf von a_2 durchführt (vgl. Abbildung 8). Hierbei wird vorausgesetzt, dass die Informationen, die a_{ref} beim Aufruf durch a_1 liefert, erst den Endpunkt des daraufhin durchgeführten Aufrufs von a_2 durch a_1 festlegt (Grounding). Dadurch ist es möglich, den Aufruf von a_1 auf ein anderes System umzuleiten, ohne dafür Anpassungen an a_1 vornehmen zu müssen. Dies setzt jedoch voraus, dass die von a_{ref} zurückgelieferte Referenz schnittstellenkompatibel zu a_2 ist.

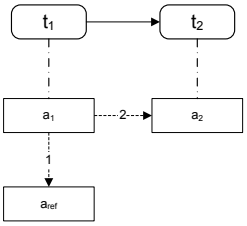


Abbildung 8: Vermittelter Aufruf

3.3.1 Fallstudie

Die Realisierung von p_3 ist in Abbildung 9 dargestellt. Das vermittelnde System ist im Falle einer SOA das Service-Verzeichnis. Erst in diesem Muster wird eine SOA (inkl. Verzeichnis) gemäß der ursprünglichen Konzeption aus [IBM00] erreicht.

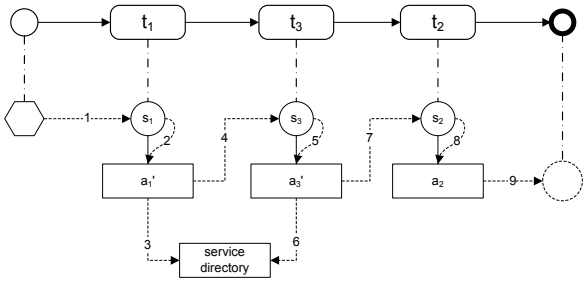


Abbildung 9: Realisierung von p_3 mit Muster *Vermittelter Aufruf*

3.3.2 Flexibilitätsbewertung

Die Wartbarkeit M_{VA} der resultierenden Architektur bei Anwendung des Musters *Vermittelter Aufruf* ist vergleichbar mit der der ersten beiden Muster ($M_{VA} = M_{DA} = M_{IA}$). Auch hier wird im Vergleich zum Muster *Direkter Aufruf* ein zusätzliches System (a_{ref}) benötigt (veringerte Wartbarkeit), womit jedoch gleichzeitig eine Explizitmachung der Prozesstransitionen von t_1 erreicht werden kann, die diesen Wartbarkeitsnachteil aufwiegt.

Der Aufwand E_{VA} zur Realisierung von P' ist in diesem Muster höher als beim Muster *Indirekter Aufruf*. Es müssen die gleichen Systeme angepasst werden wie beim Muster *Direkter Aufruf* womit $E_{VA} = E_{DA}$ gilt. Damit ergibt sich für die Flexibilität dieses Musters

$$F_{VA} = \frac{M_{VA}}{1 + E_{VA}} = \frac{M_{DA}}{1 + E_{DA}} = F_{DA}.$$

Eine höhere Flexibilität wird dann erreicht, wenn es mehrere Implementierungen ein und der selben Service-Schnittstelle gibt, die dem vermittelnden System bekannt sind. Da wir

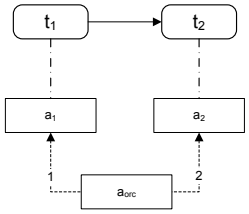


Abbildung 10: Zentraler Orchestrator

in unserem Fallbeispiel davon ausgehen, dass s_3 nicht kompatibel zu s_2 ist, folgen daraus auch bei diesem Muster die gleichen Bedingungen bzgl. Anpassungen zur Realisierung der Prozessgruppe P' wie beim Muster *Direkten Aufruf*.

Es mag an dieser Stelle überraschen, dass sich die Einführung eines Service Directory nicht positiv auf die Flexibilität auswirkt. Dies liegt daran, dass wir Flexibilität bezogen auf die Realisierung von Geschäftsprozessen untersuchen. In Bezug auf die technische Infrastruktur, bietet ein Service Directory selbstverständlich flexibilitätssteigernde Möglichkeiten, wie z.B. das Umziehen einer Funktionalität zu einer anderen technischen Komponente, ohne die Notwendigkeit zur Modifikation ihrer Klienten (lose Kopplung). Dies gilt jedoch nicht für Modifikationen an Geschäftprozessen, da es sich dabei nicht um strukturelle sondern funktionale Entitäten handelt. Solche Modifikation wirken sich auch immer auf Klienten aus. Abhilfe könnten hier semantische Methoden schaffen, die wir aber in diesem Papier nicht näher betrachten werden.

3.4 Zentraler Orchestrator

Beim Muster *Zentraler Orchestrator* besteht keine direkte Abhängigkeit mehr zwischen den task-realisierenden Systemen. Statt dessen wird davon ausgegangen, dass die Implementierung eines Tasks atomar von Systemen zur Verfügung gestellt wird und damit keinerlei prozessrelevanten Seiteneffekte beim Aufruf eines solchen Systems auftreten. Die Steuerung des Prozesses wird von einem separaten System vorgenommen, dass die Aufrufe der Systeme gemäß der vorliegenden Geschäftsprozesse koordiniert (orchestriert) (vgl. Abbildung 10).

3.4.1 Fallstudie

Dieses Muster entspricht der architekturellen Idee einer ausgereiften, prozessbefähigten SOA [KBS04]. Abbildung 11 zeigt die Realisierung von P' aus dem Fallbeispiel. Bemerkenswert hier ist, dass in dieser Realisierung keine Prozesskontextinformationen in den Systemen a_1 und a_2 mehr notwendig sind. Dies ist darin begründet, dass die Aufrufe der Services, die von diesen Systemen bereitgestellt werden, atomar sind und keine impliziten Prozesse implementieren.

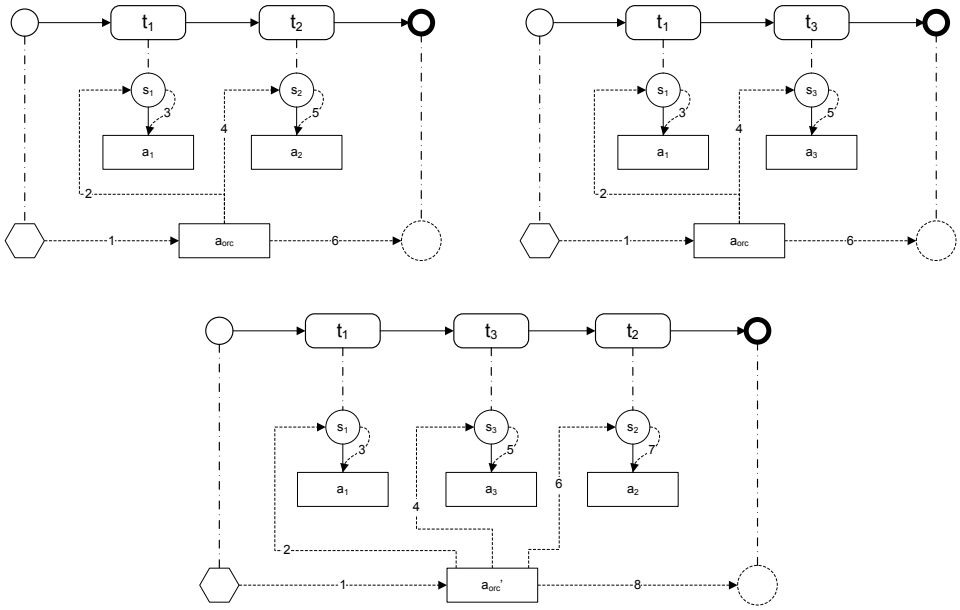


Abbildung 11: Realisierung von P' mit Muster *Zentraler Orchestrator*

3.4.2 Flexibilitätsbewertung

Durch dieses Realisierungsmuster ist die Wartbarkeit M_{ZO} der Systeme des Musters *Zentraler Orchestrator* signifikant höher als bei den vorher diskutierten Mustern ($M_{ZO} > M_{DA,IA,VA}$). Es ist nun möglich, Änderungen im Prozess zentral im System a_{orc} zu pflegen und die Systeme a_1 und a_3 unangetastet zu lassen. Die Realisierung des neuen Prozesses p_3 zieht lediglich Änderungen in a_{orc} mit sich. Innerhalb einer SOA wird a_{orc} sinnvollerweise ein System zur Pflege und zur Verwaltung von Geschäftsprozessen sein (Business Process Management, BPM), was zu einem signifikant geringerem Aufwand zur Umsetzung neuer Prozesse führt. Somit gilt für den Aufwand E_{ZO} zur Realisierung von P' mittels des Musters *Zentraler Orchestrator* $E_{ZO} < E_{IA}$ und folgerichtig der bisher beste Flexibilitätswert

$$F_{ZO} = \frac{M_{ZO}}{1 + E_{ZO}} > \frac{M_{IA}}{1 + E_{IA}} = F_{IA}.$$

3.4.3 Variante: Zentraler Orchestrator mit Vermittlung

Bei der Mustervariante *Zentraler Orchestrator mit Vermittlung* verwendet das orchestrierende System a_{orc} einen Vermittler a_{ref} zur Bestimmung des Groundings des Prozesses. Der Vorteil dieses Musters ist, dass die in a_{orc} verwendete Prozessbeschreibung zusätzlich von technischen, grounding-spezifischen Aspekten befreit werden kann, was den Prozess selbst wartbarer hält (vgl. Abbildung 12).

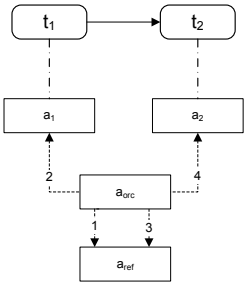


Abbildung 12: Zentraler Orchestrator mit Vermittlung

Diese Mustervariante ermöglicht ein Wechseln zu alternativen Groundings für bestimmte Tasks eines Prozesses, indem lediglich Grounding-Referenzen aus a_{ref} umgebogen werden. Die Flexibilität dieser Variante ist aufgrund der schlechteren Wartbarkeit durch Hinzunahme eines weiteren Systems etwas geringer als beim Muster *Zentraler Orchestrator*. Sie wird jedoch dann besser, wenn es für einen Service mehrere Implementierungen gibt, die je nach Prozess alternativ zueinander verwendet werden sollen.

4 Stand der Forschung

Die Reihe der Lösungsvorschläge zur Beherrschung wechselnder Geschäftsprozesse, die in bestehenden IT-Architekturen abgebildet werden müssen, reichen von Objekt-Orientierung [JEJ94] über Komponenten Frameworks [DW99] und persistenten Nachrichtensystemen [AMG95] bis hin zu den aktuellen Vorschlägen zur Verwendung von Web Services [LRS01] oder gar semantischen Web Service Architekturen [BFM02]. Auffällig hierbei ist, dass technologische Aspekte einen sehr hohen Stellenwert bei der Suche nach Lösungen einnehmen. Die Untersuchung von Prozessen und realisierenden Mustern nach formalen Aspekten ist jedoch kaum zu finden. Noch weniger findet man Belege für die Güte einer vorgeschlagenen Architektur in Form von formalen Berechnungen.

Eines der bedeutendsten Berechnungsmodelle zur Bestimmung der Flexibilität einer IT-Infrastruktur ist sicher das von Byrd und Turner [BT00]. Danach wird Flexibilität sogar als das entscheidende Maß für die Güte einer IT-Architektur herangezogen. Für unsere Betrachtungen war die Anwendung des von Byrd und Turner vorgeschlagenen Berechnungsmodells jedoch nicht geeignet, da sich dieses auf eine konkrete Instanz einer IT-Infrastruktur inklusive der Untersuchung von konkreten Werten einer spezifischen Unternehmensinstanz stützt. Wir benötigten für unsere Betrachtungen jedoch ein vereinfachtes Maß, das sich auch auf rein architekturelle Konzepte anwenden lässt. Zudem steht der Absolutheitsanspruch des aus diesem Berechnungsmodell resultierenden Flexibilitätswertes im Konflikt zu unserer Zielsetzung, lediglich die Flexibilität bezogen auf sich ändernde Geschäftsprozesse hin zu untersuchen. Dies gilt auch für das vom IEEE [IEE99] vorgeschlagenen Flexibilitätsmaß für Software.

Das von uns vorgeschlagene Berechnungsmodell lehnt sich an jene von Athey & Schmutzler [AS95] und auch Nelson & Ghods [NG98] an. Neben dem Wert für die Kosten (Aufwand) zur Realisierung einer Änderung, wurde bei unserer Berechnungsformel jedoch auch die aus der vorzunehmenden Änderung resultierende Wartbarkeit mit einbezogen.

Vergleichbar mit unserem Flexibilitätsmaß ist ebenfalls das aus dem Software-Engineering stammende Maß für die Kosten eines Evolutionsschritt von Software [EM05], aus dem die Flexibilität eines Software Systems abgeleitet werden kann. Dieses Maß haben wir auf die höhere Granularitätsstufe von Unternehmensarchitekturen gehoben und angepasst.

5 Schlussfolgerungen und Ausblick

Es wurde gezeigt, dass service-orientierte Architekturen zu einer höheren Flexibilität in Bezug auf die Realisierung einfacher Geschäftsprozesse führen können. Hierzu ist jedoch ein architektureller Entwurf gemäß des Musters *Zentraler Orchestrator* notwendig. Anhand der Muster *Direkter Aufruf*, *Indirekter Aufruf* und *Vermittelter Aufruf* wurde gezeigt, dass Services allein nicht hinreichend für die Erlangung einer erhöhten Flexibilität sind. Eine wichtige Erkenntnis der Untersuchung ist auch, dass die Einführung von Indirektionen keineswegs immer Vorteile in Bezug auf Flexibilität mit sich bringt. Abbildung 13 verdeutlicht fasst noch einmal die Werte der betrachteten Muster für die beiden flexibilitätsrelevanten Größen zusammen.

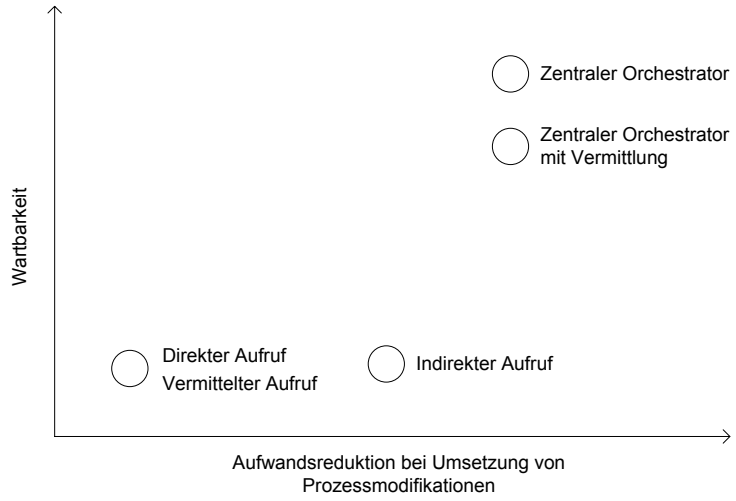


Abbildung 13: Zusammenfassung der Musterbewertung

Die von uns vorgestellten Muster erheben keinen Anspruch auf Vollständigkeit. Es sind sicherlich weitaus mehr Muster denkbar. So wäre sicherlich eine Flexibilitätsuntersuchung von Web Service Architekturen mit automatisierter Prozesskomposition unter Zuhilfenahme semantischer Methoden interessant.

Diese Arbeit legt die formale Grundlage für eine fundierte Bewertung der Flexibilität von Unternehmensarchitekturen zur Realisierung einfacher Prozesse, die lediglich aus Sequenzen besteht. Weitere Forschung ist notwendig, um eine vollständige Formalisierung der Flexibilitätsbewertung von Architekturen zur Realisierung beliebiger Prozesse durchführen zu können. In nachfolgenden Untersuchungen sollten demnach alle gängigen Muster von Workflows [vdAtHKB03], wie sie in Unternehmen zu finden sind, mit einbezogen werden.

Literatur

[AMG95] G. Alonso, C. Mohan und R. Günthör. Exotica FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In *Working Conference on Information Systems for Decentralized Organizations*, Trondheim, 1995.

[AS95] Susan Athey und Armin Schmutzler. Product and Process Flexibility in an Innovative Environment. *RAND Journal of Economics*, 26(4):557–574, Winter, 1995.

[BFM02] Christoph Bussler, Dieter Fensel und Alexander Maedche. A conceptual architecture for semantic web enabled web services. *SIGMOD Rec.*, 31(4):24–29, 2002.

[BT00] Terry Anthony Byrd und Douglas E Turner. Measuring the Flexibility of Information Technology Infrastructure: Exploratory Analysis of a Construct. *Journal of Management Information Systems*, 17(1):167–208, 2000.

[DHL01] Umeshwar Dayal, Meichun Hsu und Rivka Ladin. Business Process Coordination: State of the Art, Trends, and Open Issues. In *Proceedings of the 27th VLDB Conference5*, Roma, Italy, 2001.

[DS90] T. H. Davenport und J. E. Short. The new industrial engineering: Information technology and business process redesign. *Sloan Management review*, 31(4):11–27, 1990.

[DW99] Desmond F. D'Souza und Alan Cameron Wills. *Objects, components, and frameworks with UML: the catalysis approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[EM05] Amnon H. Eden und Tom Mens. Measuring Software Flexibility. *Technical report CSM-424*, 2005.

[Fow02] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides. *Design Patterns*. Addison-Wesley, 1994.

[Hef04] Randy Heffner. *Trends 2005: Service-Oriented Architecture And Web Services*. Forrester Research, Inc., 2004. http://www.forrester.com.

[IBM00] IBM. *Web Services architecture overview*, 2000. http://www-106.ibm.com/developerworks/webservices/library/w-ovr/.

[IEE99] IEEE. *Standard Glossary of Software Engineering Terminology 610.12-1990*, 1999.

[JEJ94] I Jacobson, M Ericsson und A Jacobsons. *The object advantage: business process reengineering with object technology*. ACM Press/Addison-Wesley Publishing, 1994.

[KBS04] Dirk Krafzig, Karl Banke und Dirk Slama. *Enterprise SOA*. Prentice Hall, 2004.

[LRS01] F. Leymann, D. Roller, und M.-T. Schmidt. Web services and business process management. *IBM Systems Journal*, (41-2), 2001.

[NG98] K. M. Nelson und M. Ghods. Measuring technology flexibility. *Eur. J. Inf. Syst.*, 7(4):232–240, 1998.

[NJFM96] T. J. Norman, N. R. Jennings, P. Faratin und E. H. Mamdani. Designing and implementing a multi-agent architecture for business process management. *Intelligent Agents III, Springer*, Seiten 261–275, 1996.

[OMG06] OMG. *Business Process Modeling Notation (BPMN) Specification*, 2006. http://www.bpmn.org/.

[PvdH06] Michael P. Papazoglou und Willem-Jan van den Heuvel. Service-Oriented Design and Development Methodology. *International Journal of Web Engineering and Technology (IJWET)*, 2006.

[STM+04] R. Schulte, J. Thompson, P. Malinverno, B. Lheureux und F. Kenney. *Predicts 2005: Application Integration, ESBs and B2B Evolve*. Gartner Research, Inc., 2004. http://www.gartner.com.

[vdAtHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski und A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, Seiten 5–51, 2003.

[W3C04] W3C. *Web Services Architecture*, 2004. http://www.w3.org/TR/ws-arch/.

[Weg03] A. Wegmann. the systemic enterprise architecture methodology, 2003.

[Wik06] Wikipedia. Flexibility — Wikipedia, The Free Encyclopedia, 2006. [http://en.wikipedia.org/w/index.php?title=Flexibility&oldid=63058403, accessed 12-July-2006].

[Zac97] John A. Zachmann. Enterprise Architecture: The Issue of the Century. *Database Programming and Design Magazine*, 1997.

[Zac99] J. A. Zachman. A Framework for Information Systems Architecture. *IBM Systems Journal*, 1999.

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Language to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER - Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods - Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahnič (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Business 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 - Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement - Erfahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometric and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 - Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14
53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de