Unterstützung von Frontloading und Look-ahead bei der Entwicklung prozessorientierter Informationssysteme in Service-orientierten Architekturen

Thomas Bauer [1], Stephan Buchwald [2]

[1] Hochschule Neu-Ulm, Fakultät Informationsmanagement
thomas.bauer@hs-neu-ulm.de
[2] T-Systems International GmbH, Delivery Unit Automotive & Manufacturing Industries
stephan.buchwald@t-systems.com

Zusammenfassung. Die in einer Service-orientierten Architektur (SOA) implementierten Geschäftsprozesse sollten durch die Fachbereiche und nicht – wie in der Praxis oftmals der Fall – durch den IT-Bereich definiert werden. Nur dann ist gewährleistet, dass das prozessorientiere Informationssystem tatsächlich den gewünschten Geschäftsnutzen realisiert. Ausführungsrelevante Prozessaspekte (z.B. Bearbeiterzuordnungen) sollten daher früh, d.h. bereits beim fachlichen Prozessentwurf, festgelegt werden (*Frontloading*). Außerdem sollten in dieser Phase die später bei der Prozessausführung zu verwendenden und bereits existierenden IT-Artefakte (z.B. Services) angegeben werden können (*Look-ahead*). Für die Festlegung solcher technischen Aspekte bieten existierende Geschäftsprozessmodellierungswerkzeuge jedoch nur wenig Unterstützung. Die Herausforderung besteht darin, dass entsprechende Ansätze für Fachanwender mit geringen IT-Kenntnissen nutzbar sein sollten, die erzeugten Informationen aber technisch eindeutig und vollständig sein müssen, damit sie für die spätere IT-Implementierung der Geschäftsprozesse genutzt werden können. Nur dann lassen sich zusätzliche Interviews in den Fachabteilungen oder Fehlinterpretationen bei der Prozessimplementierung vermeiden. Dieser Beitrag analysiert, für welche Prozessaspekte ein Frontloading bzw. Look-ahead sinnvoll ist und welche Anforderungen an die entsprechenden Modellierungstechniken bestehen. Am Beispiel von Bearbeiterzuordnungen werden Möglichkeiten zur Realisierung solcher Modellierungstechniken aufgezeigt.

Keywords: Process Engineering, Service-Orchestrierung, SOA.

1 Motivation

Eines der fundamentalen Konzepte einer Service-orientierten Architektur (SOA) ist die Orchestrierung von Geschäftsprozessen, d.h. deren koordinierte Ausführung durch eine Prozess-Engine, die Services entsprechend der definierten Prozesslogik aufruft und interaktive Arbeitsschritte (sog. *Human*

Tasks) steuert [22] [23]. Außerdem ist es Ziel einer SOA, die Durchgängigkeit bei der Umsetzung fachlicher Geschäftsprozesse in einer IT-Implementierung zu erhöhen. D.h., es ist sicherzustellen, dass der als IT-Implementierung umgesetzte Geschäftsprozess die aus fachlicher Sicht bestehenden Anforderungen auch tatsächlich erfüllt (*Business-IT-Alignment*). Diese Themen wurden im Forschungsprojekt ENPROSO (*Enhanced Process Management through Service Orientation*) betrachtet [5] [7] [8]. Bisher nicht betrachtet wurde die Fragestellung, welche Informationen ein Fachprozess-Designer dokumentieren sollte, um die spätere IT-Implementierung des Prozesses zu vereinfachen bzw. zu beschleunigen. Hierbei müssen die im Fachprozess dokumentierten Informationen ausreichend detailliert und vollständig sein, so dass ein IT-Implementierer daraus eine technische Spezifikation ableiten kann, ohne zusätzliches Wissen über diesen speziellen Fachbereich zu besitzen.

In diesem Zusammenhang bedeutet *Frontloading*, dass für Geschäftsprozesse bereits frühzeitig, d.h. während ihres fachlichen Entwurfs, detaillierte Informationen zu implementierungsrelevanten Aspekten ermittelt und dokumentiert werden (z.B. Bearbeiterzuordnungen für Human Tasks, Stellvertretungen, Ausnahmebehandlungen). Obwohl die Vorteile eines solchen Frontloading offensichtlich sind, fehlen in den in der Praxis erstellten Fachprozessmodellen entsprechende Informationen meist oder sie werden von den Fachprozess-Designern nur ungenau festgelegt. Dies wiederum führt zu Verzögerungen und Mehrkosten bei der nachfolgenden Prozessimplementierung, weil eine zusätzliche Analysephase notwendig wird. Hierbei müssen (erneut) Interviews mit Fachverantwortlichen bzw. -anwendern geführt werden, und das in einer Projektphase in der diese Personen sowie die entsprechenden Projektberater oft nicht mehr zur Verfügung stehen. Um Projektverzögerungen zu vermeiden, wird für solche ungenau definierten Prozessaspekte oftmals auch vom IT-Bereich selbst entschieden, wie die Umsetzung im Detail zu gestalten ist. Dies kann wiederum zu einer fehlerhaften Prozessimplementierung führen, also die *Prozessqualität* [19] [20] beeinträchtigen.

Es stellt sich die Frage, warum Fachprozess-Designer die später benötigten Informationen nur unvollständig oder ungenau dokumentieren. Ein Problem liegt darin, dass Fachanwender bzw. -berater oftmals nicht wissen, welche Prozessaspekte in welchem Detaillierungsgrad für eine spätere Prozessimplementierung benötigt werden. Auch die in Unternehmen verwendete SOA-Methodik [1] [11] [13] [23] geht hierauf nur unzureichend ein (vgl. Abschnitt 4). Außerdem haben Fachprozess-Designer häufig nicht die erforderlichen Kompetenzen, um Prozessaspekte auf einer detaillierten technischen Ebene zu beschreiben. Existierende *Prozessmodellierungssprachen* (z.B. erweiterte Ereignis-Prozess-Ketten (eEPK), Business Process Modeling Notation (BPMN)) und Geschäftsprozessmodellierungswerkzeuge (z.B. ARIS) bieten keine für diesen Personenkreis verwendbaren Techniken, um technische

Sachverhalte ausreichend detailliert zu dokumentieren. So kann z.B. bei einer ARIS eEPK kaum modelliert werden, dass ein Prozessschritt Y (z.B. Ermittlung der von einer geplanten Produktänderung betroffenen Bauteile) von einer Person mit Rolle „Hardware-Entwickler“ ausgeführt werden soll, die zudem zur selben Abteilung gehört, wie der Bearbeiter eines vorangehenden Prozessschrittes X („Produktänderungsantrag erstellen“). Meist wird für Prozessschritt Y dann nur die Rolle modelliert oder es wird zusätzlich ein Abteilungsobjekt mit unklarer Semantik verwendet (vgl. Abb. 1a). Diese Information ist aber nicht ausreichend, um den Prozess so implementieren zu können, dass er tatsächlich von einer Prozess-Engine in der gewünschten Semantik ausgeführt werden kann. Es fehlt die Information, aus welcher Abteilung der Bearbeiter stammen soll. In Modellierungswerkzeugen wie ARIS ist es jedoch nicht oder nur eingeschränkt möglich, die in Abb. 1a gestrichelt dargestellten Abhängigkeiten direkt zu modellieren. So gibt es kein Objekt, das denjenigen Bearbeiter repräsentiert, der einen bestimmten Prozessschritt tatsächlich bearbeitet. Ebenso gibt es keine Beziehung, die ausdrückt, dass die an einem Prozessschritt modellierte Abteilung dieselbe sein soll, wie die Abteilung, welcher dieser tatsächliche Bearbeiter angehört.

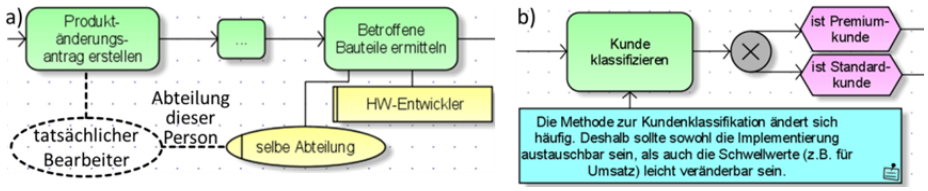

Abb. 1. ARIS eEPK mit a) unvollständiger Bearbeiterzuordnung, b) Flexibilitätsmarkierung.

Ähnliche Fragestellungen ergeben sich im Zusammenhang mit *Look-ahead*: Meist werden Geschäftsprozesse ohne detaillierte Bezugnahme auf die bereits existierenden IT-Artefakte modelliert. Jedoch ist es aus Sicht der Prozessimplementierer wünschenswert, dass die bei der Implementierung zu verwendenden und bereits vorliegenden technische Artefakte (z.B. Services, Objekte des Organisationsmodells) beim Fachprozess-Design ausgewählt und referenziert werden. Dies zu bewerkstelligen, ist für Fachprozess-Designer aber nicht trivial, da sie mit IT-lastigen Beschreibungen technischer Artefakte kaum klar kommen. So werden Services, die durch Geschäftsprozesse aufgerufen werden sollen, oftmals mittels der *Web-Service Definition Language* (WSDL) beschrieben. WSDL-Artefakte sind aber für Personen ohne entsprechenden IT-Hintergrund kaum verständlich. Selbst die von einigen IT-Bereichen angebotenen textuellen Service-Beschreibungen sind für Fachanwender nur bedingt verwendbar, da sie technische Terme enthalten oder technische Doku-

mentationen referenzieren (z.B. XSD-Beschreibungen für Datentypen der Ein-/Ausgabedaten von Service-Aufrufen).

Für beide Aspekte, Frontloading und Look-ahead, werden Techniken benötigt, die es Fachprozess-Designern ohne tiefgehende IT-Kenntnisse erlauben, technische Aspekte zu definieren bzw. zu referenzieren. Dann wird es möglich, die Verwendung dieser Techniken in der SOA-Methodik des Unternehmens zu verankern und durch SOA-Governance sicherzustellen. Abschnitt 2 betrachtet, für welche Aspekte eines Geschäftsprozesses ein *Frontloading* relevant ist. Außerdem werden die resultierenden Anforderungen beschrieben und deren Umsetzung am Beispiel von Bearbeiterzuordnungen exemplarisch vorgestellt. In Abschnitt 3 werden die Prozessaspekte mit Bedarf für ein *Look-ahead* diskutiert. Eine Diskussion der einschlägigen Literatur und relevanter SOA-Methodiken erfolgt in Abschnitt 4, bevor der Beitrag mit einer Zusammenfassung und einem Ausblick schließt.

2 Frontloading

In diesem Abschnitt analysieren wir, für welche Prozessaspekte ein *Frontloading* relevant ist und welche Anforderungen hierfür bestehen. Anschließend wird für Bearbeiterzuordnungen im Kontext von Human-Tasks exemplarisch aufgezeigt, wie sich Frontloading konkret realisieren lässt (weitere Anwendungen finden sich in [8]).

2.1 Relevante Prozessaspekte

Zuerst einmal geht es darum, besser zu verstehen, welcher Zweck mittels Frontloading verfolgt werden soll. Dazu betrachten wir verschiedene Aspekte eines Geschäftsprozesses [23] und analysieren, *welche* Informationen hierzu bereits in der frühen Phase des Fachprozess-Designs dokumentiert werden sollten.

Organisatorischer Aspekt: In einem Geschäftsprozessmodell wird mittels sog. *Bearbeiterzuordnungen* festgelegt, welche Personen einen bestimmten Prozessschritt bearbeiten sollen. Diese Bearbeiterzuordnungen werden in einer späteren Phase des Entwicklungszyklus in einem Prozess-Management-System technisch umgesetzt [15] [25] [26] [27]. Die zugehörige Prozess-Engine sorgt dann zur Laufzeit dafür, dass der Prozessschritt tatsächlich exakt diesen Personen in die Arbeitsliste eingestellt wird, so dass nur diese ihn bearbeiten können. Daher ist die exakte fachliche Modellierung von Bearbeiterzuordnungen eine wichtige Grundlage für die Implementierungsphase.

Im Beispiel aus Abb. 1 muss im Fachprozessmodell die Bearbeiterzuordnung

Rolle = Hardware-Entwickler ∧
Abteilung = Abteilung(Bearbeiter(Produktänderungsantrag erstellen))

festgelegt werden. Auf Grundlage der Teilinformation *Rolle = Hardware-Entwickler* (vgl. Abb. 1a) allein ist dagegen keine vollständige bzw. korrekte Implementierung möglich, da der resultierende Bearbeiterkreis (mit allen Hardware-Entwicklern) zu groß wäre. Wie sich mit beschränkten IT-Kompetenzen dennoch verwendbare Bearbeiterzuordnungen modellieren lassen zeigt Abschnitt 2.3.

Zum organisatorischen Aspekt eines Geschäftsprozesses gehören auch *Stellvertreterregelungen* [2]. Diese legen für den Fall der Abwesenheit regulärer Bearbeiter (z.B. infolge von Dienstreisen, Urlaub oder Krankheit) fest, welche Personen einen Prozessschritt alternativ bearbeiten können. Auch diese Funktionalität wird von heutigen Prozess-Engines bereitgestellt. Sie sollte auch genutzt werden, da sich Prozessschritte sonst ggf. nur in den Arbeitslisten abwesender Personen befinden und daher nicht bearbeitet werden können. Dies kann zu signifikanten Verzögerungen bei der Prozessausführung und damit zu hohen Kosten (z.B. infolge versäumter Termine) führen. Um dies zu vermeiden, müssen geeignete Stellvertreterregelungen für alle Geschäftsprozesse bzw. Prozessschritte implementiert werden. Dies ist wiederum nur möglich, wenn basierend auf den fachlichen Anforderungen vorgegeben worden ist, welcher Personenkreis als Stellvertreter fungieren kann bzw. soll. Auch für diese Art des *Frontloading* können die in Abschnitt 2.3 beschriebenen Techniken zur fachlichen Modellierung von Bearbeiterzuordnungen verwendet werden.

Ähnliches gilt für *Eskalationsregeln:* Diese definieren z.B. Mitteilungen an bestimmte Kollegen oder Vorgesetzte, die ausgelöst werden sollen, wenn die Bearbeitung eines Prozessschrittes nicht fristgerecht begonnen oder beendet wird [18] [23]. Auch hier ist wieder, ausgehend von den fachlichen Anforderungen, vorzugeben, in welchen Fällen, nach welchen Fristen, an welche Personen, welche Art von Eskalation ausgelöst werden soll. Eine eigenmächtige Festlegung durch die IT-Implementierer ist auch hier nicht sinnvoll, insbesondere wenn man berücksichtigt, dass eine IT-Implementierung häufig durch externe Dienstleister erfolgt. Diese haben nur begrenzte Möglichkeiten und kaum wirtschaftliches Interesse, unklare Sachverhalte mit dem betroffenen Fachbereich des Auftraggebers zu klären. Insgesamt ist deshalb das Frontloading für alle Themen des organisatorischen Aspekts essentiell.

Datenobjekte: Fachliche Datenobjekttypen werden in einem Prozessmodell benötigt, um die Schnittstellen des Prozesses zu definieren und seine internen Datenflüsse festzulegen [17] [23]. So werden Eingabedaten einem Prozess beim Start übergeben, während er bei seinem Ende entsprechende Ausgabedaten zurückliefert. Des Weiteren müssen die Ein- und Ausgabedaten der vom

Prozess zur Laufzeit aufgerufenen Services verwaltet werden. Im Kontext einer *Human Task* etwa repräsentieren die Eingabedaten die Informationen, welche dem Bearbeiter in seiner Bildschirmmaske angezeigt werden, während die Ausgabedaten den von ihm einzugebenden Daten entsprechenden [14].

Werden Datenobjekttypen in Fachmodellen nur rudimentär modelliert, etwa durch Angabe ihres Namens und ihrer groben Inhalte, reicht diese Information nicht aus, um die Aufruf-Schnittstelle des Prozesses oder die von ihm benötigten Services (inkl. *Human Tasks* und ihrer Bildschirmmasken) zu implementieren. Um mittels *Frontloading* spätere Rückfragen bei den Fachabteilungen reduzieren zu können, müssen für jedes Datenobjekt alle fachlich relevanten Attribute (inkl. Ihrer Datentypen) modelliert werden. So muss der Fachbereich entscheiden, ob das Attribut *Kundennummer* im Datenobjekt *Kunde* eine Zahl sein soll, oder ob *Kundennummern* der Form „K285-2012“ erwünscht sind. Außerdem müssen die Datenobjekte den Geschäftsprozessschritten bzw. fachlichen Services eindeutig als Eingabe- oder Ausgabeparameter zugeordnet werden.

Services: Ein automatisch ausgeführter Prozessschritt kann einen Service aufrufen, ihm Datenobjekte übergeben und nach dessen Ausführung die Ausgabedaten zurückerhalten, die dann wiederum die Prozessdaten verändern. *Frontloading* fordert hier, dass die benötigten Services bereits im fachlichen Prozessmodell möglichst detailliert beschrieben werden und nicht nur ein Servicename als Platzhalter modelliert wird. Erst dadurch wird definiert, wie der zusätzlich benötigte Service gestaltet werden soll.

Konkret sollte für jeden zu verwendenden Service festgelegt werden, welche Funktion er erbringt, welche Ein-/Ausgabedaten er verwendet und welche QoS-Eigenschaften [5] erforderlich sind (z.B. Antwortzeit, Transaktionsverhalten). Dies kann vom IT-Bereich mit existierenden Services abgeglichen werden oder als Entwurfsgrundlage für einen neu zu realisierenden Service dienen.

Flexibilität: Für eine IT-Umsetzung ebenfalls relevant ist die Markierung der Stellen im Geschäftsprozess, für die ein (erhöhter) Flexibilitätsbedarf besteht [9] [10] [21] [23] [33]. Für diese Bereiche oder Objekte im Geschäftsprozess ist eine besonders flexible Implementierung erforderlich. So gibt es Prozessschritte, die einen Service aufrufen, bei denen aber bekannt ist, dass der konkret zu verwendende Service variiert. So kann die Überprüfung einer Kreditwürdigkeit mittels firmeninterner Informationen und IT-Systemen erfolgen oder von externen Dienstanbietern erbracht werden, die sich in Qualität und Preis unterscheiden. Zudem können die Ein-/Ausgabedatenstrukturen unterschiedlich sein. Ist dem Fachbereich z.B. durch Erfahrungen aus der Vergangenheit bekannt, dass der zu verwendende Service sich häufig ändert, sollte dieser Prozessschritt entsprechend markiert werden. Dadurch wird es möglich,

den Serviceaufruf in besonderem Maße entkoppelt zu implementieren, etwa durch Verwendung eines *Enterprise Service Bus* (ESB) oder *Message Brokers*.

Ein vorhersehbarer Bedarf an Flexibilität besteht auch im Hinblick auf die Festlegung von Verzweigungsbedingungen, Geschäftsregeln, Bearbeiterzuordnungen und Timeout-Zeiten. Dies sollte deshalb bereits beim Fachentwurf dokumentiert werden. Hierfür genügt es meist, im Modellierungswerkzeug einen speziellen Objekttypen für Flexibilitätskommentare zu definieren bzw. von einem existierenden Objekttyp abzuleiten (vgl. Abb. 1b). Dieser muss lediglich ein Attribut zur Angabe der Flexibilitätsart besitzen sowie eine textuelle Beschreibung der zu erwartenden Veränderungen ermöglichen. Die Anforderungen des in Abb. 1b dargestellten Beispiels können z.B. durch Einbindung einer *Business Rule Engine* realisiert werden.

Ausnahmesituationen: Bei der Ausführung von Geschäftsprozessen können Ausnahmesituationen auftreten, auf die geeignet reagiert werden muss [23]. Beispiele für Ausnahmesituationen sind fehlgeschlagene Serviceaufrufe, unzureichende Datenqualität (d.h. fehlende oder widersprüchliche Attributwerte, etwa nach einer Dateneingabe durch einen Benutzer) oder eine leere Bearbeitermenge für eine Human-Task (etwa wenn Mitarbeiter das Unternehmen verlassen haben). Für einige dieser Ausnahmesituationen ist es offensichtlich, unter welchen Umständen sie auftreten können (z.B. Fehlschlag eines Serviceaufrufs), so dass die Detektion und Behandlung der Ausnahme direkt durch den IT-Bereich definiert werden kann, d.h. für diesen Fall besteht kein Bedarf für ein Frontloading. Dagegen sind andere Ausnahmen nur mittels fachlicher Zusatzinformationen erkennbar, etwa widersprüchliche Attributwerte (z.B. *Baureihe = Actros* und *Bauteil = P2354-2356-3*). Zur Fehlererkennung muss bekannt sein, dass ein Actros ein Nutzfahrzeug ist, wohingegen mit P beginnende Bauteilnummern zu PKWs gehören.

Während die Ausnahmenerkennung nur in manchen Fällen vom Fachbereich vordefiniert werden muss, ist Frontloading für die Ausnahmebehandlung fast immer erforderlich: So kann ein Serviceaufruf nach einem Fehlschlag automatisch wiederholt werden. Schlägt er jedoch weiterhin fehl, muss vom Fachbereich vorgegeben sein, ob der Aufruf weiterhin in gewissen Zeitintervallen wiederholt werden soll, ob ein alternativer Service aufgerufen werden soll (ggf. mit höheren Kosten), oder ob von Personen ausgeführte *Human Tasks* stattdessen verwendet werden sollen, um die Serviceergebnisse auf diesem Weg zu erzeugen. Hierbei sind u.a. die resultierenden Verzögerungen bei der Prozessausführung gegen die entstehenden Zusatzkosten abzuwägen. Ebenso muss bei Inkonsistenzen in den Daten definiert werden, wer den Fehler beheben soll. Im Falle fehlender Task-Bearbeiter sollte ein Prozessverantwortlicher festgelegt werden, der den Prozessschritt dann an eine fachlich kompetente Person delegieren kann.

Bildschirmmasken: Oftmals verwenden Benutzer als Formulare gestaltete Bildschirmmasken zur Bearbeitung von *Human Tasks* [14] [16] [17]. Die anzeigbaren bzw. einzugebenden Daten werden dabei durch die Ein-/Ausgabedatenobjekte des entsprechenden Prozessschrittes bestimmt. Allerdings sollten dem Benutzer nicht alle möglichen Attribute aus dem Eingabeobjekt angezeigt werden, wenn dieses redundante, irrelevante oder technische (d.h. für Fachanwender uninteressante) Attribute enthält. Ebenso sollte der Benutzer nicht alle theoretisch möglichen Daten eingeben, nur weil die als Ausgabeobjekt verwendete Standard-Datenstruktur im aktuellen Kontext irrelevante Attribute enthält. Solche Sachverhalte sind durch den Fachbereich vorzugeben, um die spätere Implementierung von möglichst gut geeigneten Bildschirmmasken zu ermöglichen.

Datenobjekttypen enthalten keine Informationen darüber, welche Attribute obligat und welche optional sind. Daher kann auf Basis der als Ausgabeobjekt verwendeten Datentypen nicht entschieden werden, ob die Eingabe eines bestimmten Attributs in einem Pflichtfeld erfolgen soll oder nicht. Die Maskenimplementierung kann dies aber nur berücksichtigen, wenn der Fachbereich beim betreffenden Prozessschritt entsprechende Information explizit dokumentiert hat. Es sollte für jedes Attribut des Masken-Ausgabeobjektes ein Status $\in$ {*verpflichtend*, *optional*, *weglassen*} definiert werden sowie für Attribute des Eingabeobjektes ein Status $\in$ {*anzeigen*, *weglassen*}.

Neben dem Inhalt von Bildschirmmasken ist auch deren Gestaltung relevant. Ein Frontloading sollte daher auch Informationen zum Aussehen der Masken beinhalten. So ist es für die spätere Maskenimplementierung hilfreich, wenn am Prozessschritt eine Skizze angehängt wird, welche die Positionen der einzelnen Datenfelder fixiert und die darzustellenden Texte-Labels festlegt. Außerdem kann vorgegeben werden, mit welcher Art von Steuerelement eine Werteeingabe erfolgen soll (z.B. Textfeld, Listenfeld, Combo-Box, Radio-Buttons).

Sonstiges: Bzgl. weiterer Prozessaspekte, etwa der Definition von Transaktionsgrenzen im Geschäftsprozess oder der Festlegung von Messpunkten für ein späteres Prozess-Performance-Management, verweisen wir aus Platzgründen auf [8].

2.2 Anforderungen an Frontloading

Frontloading-Techniken müssen die folgenden generellen Anforderungen erfüllen:

- Es muss eine ***Modellierungstechnik für jeden Prozessaspekt*** geben, wenn dessen Ausgestaltung bei der Prozessimplementierung aus fachlicher Sicht relevant ist.

- Jede Technik zum Frontloading muss auch mit geringen IT-Kenntnissen ***leicht verständlich und benutzbar*** sein.
- Der resultierende ***Informationsgehalt bildet eine vollständige Grundlage*** für eine spätere Prozessimplementierung.
- Die ***Überführung der Informationen in eine Prozessimplementierung*** ist möglich und sollte möglichst effizient durchführbar sein.

2.3 Frontloading für Bearbeiterzuordnungen

Wir stellen im Folgenden vier Beschreibungstechniken für Bearbeiterzuordnungen vor. Diese erfüllen die genannten Anforderungen und bieten in der dargestellten Reihenfolge aufsteigenden Informationsgehalt und Funktionalität, bei gleichzeitig zunehmendem Bedarf nach einer Unterstützung durch Modellierungswerkzeuge.

Ansatz 1 (Freie textuelle Beschreibung): Eine heute bereits verwendbare Beschreibungstechnik ist die freie Dokumentation am zugehörigen Prozessschritt. Hierfür können vom Modellierungswerkzeug angebotene Objekte für Anmerkungen (vgl. Abb. 1b), Attribute des Prozessschrittes oder angehängte Textdokumente genutzt werden. So könnte z.B. bei dem in Abb. 1a dargestellten Beispiel am Prozessschritt „Betroffene Bauteile ermitteln“ Folgendes dokumentiert werden: *„Bearbeiter muss Rolle Entwickler haben und gleicher Abteilung angehören, wie der Antragssteller“*.

Prinzipiell ist so ein hoher Informationsgehalt erzielbar, zudem erfordert die Beschreibungstechnik keine speziellen Fähigkeiten der Fachprozess-Designer. Allerdings lässt sich die Qualität solcher Beschreibungen kaum sicherstellen, da diese ungenau und mehrdeutig sein können. So muss vor der Überführung in eine IT-Implementierung festgestellt werden, welche Objekte des Organisationsmodells (z.B. Rollennamen) genau gemeint sind. Dies ist nicht immer einfach, da sich Fachprozess-Designer oftmals nicht an Standard-Begriffe halten (z.B. Entwickler vs. Hardware-Entwickler). Auch die Verknüpfung der Teilbedingungen (und/oder sowie Klammerung) und die Referenzierung von Prozessschritten wird oft nicht eindeutig sein. Zudem entsteht für die manuell durchzuführende Interpretation ein hoher Aufwand.

Ansatz 2 (Semantische Beschreibung): Eine Erweiterung von Ansatz 1 besteht darin, für Objektnamen ein vordefiniertes Vokabular (im Sinne einer Ontologie) vorzuschreiben. So kann der Fachprozess-Designer z. B. verpflichtet werden, bei der Verwendung eines Rollennamens einen vordefinierten Namen aus einer gegebenen Liste auszuwählen. Ansonsten darf er die Beschreibung der Bearbeiterzuordnung frei gestalten. Eine weitergehende Variante besteht darin, den erstellten Text automatisiert zu analysieren, um Referenzierungen des Organisationsmodells zu identifizieren und mit dem vorde-

finierten Vokabular abzugleichen. So wird im oben dargestellten Text erkannt, dass es sich bei dem Wort „Entwickler“ um einen Rollennamen handeln soll. Ergibt der Abgleich mit dem vordefinierten Vokabular, dass es keine Rolle „Entwickler“ gibt, sondern nur „Software-Entwickler“ und „Hardware-Entwickler“, wird der Fachprozess-Designer aufgefordert, seine Angabe zu präzisieren. Im dargestellten Beispiel einer Bauteil-Ermittlung wird er die Rolle „Hardware-Entwickler“ wählen.

Mit diesem Ansatz wird die Ungenauigkeit bzw. Unschärfe verringert, da die Objektnamen nun eindeutig sind. Insofern reduziert sich auch der Aufwand für die Überführung in eine Prozessimplementierung. Allerdings ist die Interpretation des frei formulierten Textes, ebenso wie bei Ansatz 1, erforderlich.

Ansatz 3 (Template-basierte Beschreibung): Eine weitere Optimierung kann dadurch erzielt werden, dass dem Fachprozess-Designer, neben den Objektnamen, auch die Beschreibungstexte vorgegeben werden. So kann die Menge aller relevanten Bearbeiter-Schablonen (engl. *Templates*) vordefiniert und bereitgestellt werden. Abb. 2 zeigt einige Beispiele für solche *Templates*.

Template-Name	**Beschreibung**
Rolle(x)	Die potentiellen Bearbeitern müssen die Rolle x innehaben
Abteilung(x)	Bearbeiter sind Mitglied der Abteilung x
Kompetenz(x)	Bearbeiter haben Kompetenz x (z.B. Englisch)
Vorgesetzter(x)	Bearbeiter ist Vorgesetzter von Person x
RolleUndAbt(x,y)	Bearbeiter haben die Rolle x und sind Mitglied der Abteilung y

Abb. 2. Beispiel-Templates für Bearbeiterzuordnungen.

Der Fachprozess-Designer wählt bei der Festlegung einer Bearbeiterzuordnung ein Template aus und gibt die Namen der referenzierten Objekte an (anstatt x, y). Um Mehrdeutigkeiten zu vermeiden, sollten diese Namen analog zu Ansatz 2 aus einem vordefinierten Vokabular entnommen werden. In unserem Beispiel wird der Fachprozess-Designer also das Template *RolleUndAbt(x,y)* auswählen und als Rolle x = „Hardware-Entwickler“ sowie als Abteilung y = „Abteilung von Bearbeiter(Schritt Produktänderungsantrag erstellen)“ festlegen.

Mit diesem Ansatz lässt sich eine hohe Genauigkeit erzielen. Lediglich die Referenzierung von Vorgängerschritten oder Prozessvariablen ist evtl. noch nicht eindeutig. Dies kann aber durch eine geeignete Unterstützung seitens des Modellierungswerkzeugs verbessert werden. Es besteht also in Bezug auf eine weitere Reduzierung des Aufwandes für nachfolgende Entwicklungsphasen kaum mehr Optimierungspotential. Nachteilig bei Ansatz 3 ist allerdings, dass sehr viele Kombinationen von Bedingungen als Template vordefiniert werden müssen (z.B. *RolleUndAbteilung, RolleUndKompetenz, Abtei-*

lungUndGruppe …). Diese Problematik greift der nun vorgestellte Ansatz 4 auf.

Ansatz 4 (Kombinierbare Templates) Werden dem Fachprozess-Designer lediglich elementare Templates angeboten, muss er diese zur Erstellung von komplexeren Bearbeiterzuordnungen mittels boolescher Operationen (AND, OR, NOT) kombinieren. Inwieweit er hierzu überhaupt in der Lage ist, hängt von seinem (mathematischen) Kenntnisstand ab. Schwer vorstellbar ist, dass er komplexe Formeln selbstständig korrekt erstellt. Deshalb benötigt ein Fachprozess-Designer bei diesem Ansatz eine Unterstützung durch ein geeignetes Modellierungswerkzeug zur Festlegung von Bearbeiterzuordnungen. Dieses ermöglicht die Auswahl eines (elementaren) Templates (vgl. Abb. 3a) sowie von Objektnamen aus dem vordefinierten Vokabular. Des Weiteren können die Namen der existierenden Prozessschritte referenziert werden. Nach Festlegung einer ersten elementaren Bearbeiterzuordnung, ermöglicht das Werkzeug dann die Hinzunahme weiterer Teile. Nach Festlegung jedes weiteren Teils muss darüber hinaus die Verknüpfungsart angeben werden. Ihre Bedeutung sollte, wie in Abb. 3b dargestellt, vom Modellierungswerkzeug geeignet erläutert werden.

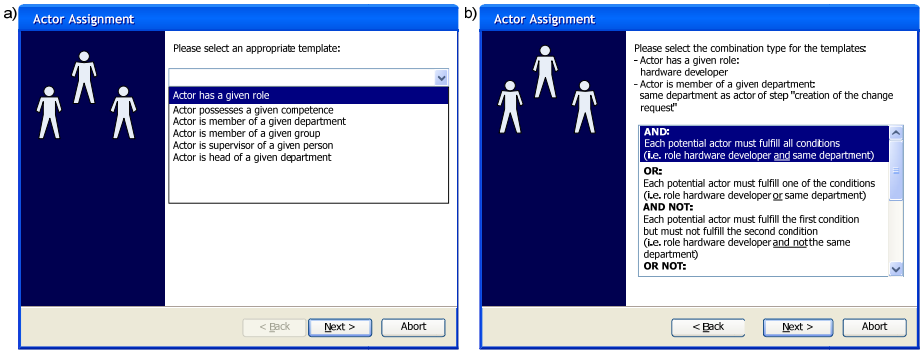

Abb. 3. Assistent für a) die Auswahl elementarer Templates, b) und deren Kombination.

Die beliebige Kombinierbarkeit von Templates steigert die Ausdrucksmächtigkeit für Bearbeiterzuordnungen. Deren Festlegung wird für Fachprozess-Designer allerdings auch anspruchsvoller. Bzgl. der Genauigkeit der erstellten Bearbeiterzuordnungen sowie dem Aufwand für die Überführung in eine Prozessimplementierung besteht kein nennenswerter Unterschied zu Ansatz 3. Deshalb sollte man die Entscheidung zwischen den Ansätzen 3 und 4 davon abhängig machen, wie gut die Fachprozess-Designer mit Ansatz 4 zurechtkommen.

3 Look-ahead

Wir betrachten nun die einzelnen für Look-ahead relevanten Prozessaspekte sowie zugehörige allgemeinen Anforderungen. Eine detaillierte Beschreibung von Ansätzen zur Realisierung von Look-ahead für konkrete Prozessaspekte findet sich in [8].

3.1 Relevante Prozessaspekte

Für die folgenden Prozessaspekte sollte bereits bei der Modellierung des Fachprozesses berücksichtigt werden, ob bzw. welche IT-Artefakte bereits existieren (oder in Entwicklung sind) und später bei der Prozessausführung verwendet werden sollen.

Organisatorischer Aspekt: Wie bereits im Kontext von Frontloading erörtert, beschreibt ein Fachprozessmodell auch Bearbeiterzuordnungen, Stellvertreterregelungen und Eskalationen. Diese werden mittels Ausdrücken definiert, die organisatorische Objekte (kurz: OrgObjekte), etwas Rollen, Abteilungen oder Kompetenzen, verwenden. Bei der späteren Prozessimplementierung werden diese Ausdrücke so umgesetzt, dass Objekte eines Organisationsmodells des Unternehmens referenziert werden. Damit dieses bei der Prozessausführung auch tatsächlich zur Berechnung der entsprechenden Personen (potentielle Bearbeiter) verwendet werden kann, enthält es außer den OrgObjekten auch die jeweils aktuell gültige Zuordnung von Personen zu ihnen. So muss z.B. gespeichert sein, welche Mitarbeiter aktuell die Rolle *Hardware-Entwickler* innehaben, um Bearbeiter einer *Human Task* mit entsprechender Rollenzuordnung ermitteln zu können.

Damit ein Fachprozess in eine durch eine Prozess-Engine ausführbare Implementierung umgesetzt werden kann, müssen die zu verwendenden OrgObjekte eindeutig identifiziert werden können. Idealerweise werden hierzu im Fachprozess dieselben Namen für OrgObjekte verwendet, wie im Organisationsmodell des Unternehmens.[1] Dann ist eine einfache Umsetzung bei der Prozessimplementierung gewährleistet.

Allerdings handelt es sich bei OrgObjekten um IT-Artefakte (wenn auch um einfach strukturierte), die den Fachprozess-Designern nicht ohne weiteres bekannt sind. Um sie für Look-ahead verwendbar zu machen, müssen sie diesem Personenkreis zugänglich gemacht werden. Dies erfordert einerseits eine leicht verständliche Publizierung der Organisationsmodell-Struktur (z.B. der Objekttypen Person, Rolle, Abteilung und Kompetenz sowie ihrer Beziehungen). Andererseits müssen die Ausprägungen der OrgObjekte publiziert wer-

[1] Durch das in Abschnitt 2.3 erwähnte vordefinierte Vokabular wird dies unterstützt, so dass die vorgestellten Ansätze teilweise auch Look-ahead realisieren.

den, d.h. die konkreten im Prozessmodell verwendbaren Rollen, Abteilungen, usw. Schließlich müssen geeignete Suchfunktionen bereitgestellt werden.

Offensichtlich erhöht Look-ahead die Präzision der Fachprozessmodelle und reduziert zudem Mehrdeutigkeiten. Die zusätzlich resultierende Wiederverwendung der OrgObjekte erspart darüber hinaus deren (redundante) Festlegung. Noch viel entscheidender ist jedoch, dass durch die Vermeidung identischer bzw. ähnlicher Objekte der Pflegeaufwand reduziert wird: So müssen Personen bei einem Bereichs- oder Funktionswechsel nicht mehreren ähnlichen OrgObjekten (z.B. Rollen) neu zugewiesen (und aus ihren ehemaligen OrgObjekten herausgenommen) werden. Um den Pflegeaufwand weiter zu reduzieren, lohnt es sich sogar, nur „ungefähr passende“ OrgObjekte in einem Geschäftsprozess zu verwenden und sie mittels einer geeigneten Bearbeiterzuordnung anzupassen. So könnte z.B. anstatt einer noch nicht existierenden Rolle *Software-Ersteller* die äquivalente Bearbeiterzuordnung Rolle = *Software-Entwickler* $\vee$ Rolle = *Software-Architekt* verwendet werden, falls diese beiden OrgObjekte bereits existieren.

Datenobjekte: In einem Fachprozess werden Datenobjekte benötigt, um den Datenfluss innerhalb des Prozesses sowie beim Aufruf von Services oder *Human Tasks* festzulegen. Look-ahead bedeutet nun, dass hierfür bereits existierende Typen von Datenobjekten verwendet werden sollten, d.h. Datentypen für die bereits ein detailliertes Modell (z.B. als UML-Klassendiagramm) oder gar eine Implementierung (z.B. als Java-Klasse) existiert. Wie bereits motiviert, ist es dazu erforderlich, dass die Informationen über existierende technische Artefakte auf eine für Fachprozess-Designer verständliche Art bereitgestellt werden.

Der offensichtliche Nutzen einer Wiederverwendung technischer Datentypen ist die Reduzierung des Aufwandes für deren Modellierung, Spezifikation und Implementierung. Noch wichtiger ist jedoch, dass durch Look-ahead erreicht wird, dass innerhalb eines Geschäftsprozesses dieselben Datentypen verwendet werden, wie für Serviceaufrufe und *Human Tasks*. Eine solche Harmonisierung erspart auch den Aufwand für die aufwendige Implementierung von Transformationen zwischen Datenstrukturen. Andernfalls müssten z.B. Kundendaten vor einem Serviceaufruf von dem prozessinternen Datentyp in den vom Service verwendeten Datentyp transformiert werden; ebenso müssten (z.B. nach Änderung der Kundendaten durch den Service) die Rückgabedaten des Service wieder in die prozessinterne Datenstruktur zurücktransformiert werden.

Services: Ein Geschäftsprozess ruft bei seiner Ausführung Services auf, um Aktionen in (fremden) IT-Systemen durchzuführen. Die in einem Unternehmen existierenden IT-Systeme bieten hierzu die relevanten Aktionen in Form einer Service-Schnittstelle an. Wie bereits in Abschnitt 1 erwähnt, werden die

angebotenen Services und ihre Operationen häufig lediglich auf einer sehr technischen Ebene publiziert, etwas als WSDL-Beschreibungen. Diese sind jedoch für Fachprozess-Designer mit geringen IT-Kenntnissen nicht verständlich. Dasselbe Problem besteht auch bei den von manchen IT-Bereichen angebotenen textuellen Servicebeschreibungen, wenn die verwendete Sprache sich zu stark von der fachlichen Ebene unterscheidet oder gar technische Artefakte wie XSD-Datentypspezifikationen für Ein-/Ausgabeparameter referenziert werden. Daher ist es erforderlich, dass existierende und bereitgestellte Services in einer für Fachanwender verständlichen Sprache bzw. Notation beschrieben werden. Diese Beschreibungen müssen leicht zugänglich sein und das Suchen nach benötigten Services ermöglichen. Nur dann wird es möglich, dass Fachprozess-Designer einen geeigneten Service finden und am entsprechenden Geschäftsprozessschritt referenzieren können. Eine solche präzise Festlegung eines Services (möglichst inkl. der zu verwendenden Serviceoperation) erspart aufwendige Rückfragen und Analysen während der Implementierungsphase des Geschäftsprozesses.

Nicht immer existiert jedoch bereits ein Service, der exakt die gewünschte Funktionalität realisiert. Es macht aber in manchen Fällen Sinn, einen Service zu verwenden, der eine ähnliche Funktionalität anbietet. Gibt es beispielsweise keinen Service, der nach Übergabe eines Bauteilnamens die Bauteildaten bereitstellt, so kann dieser ersetzt werden durch einen bereits existierenden Service, der die Bauteilnummer als Eingabe verwendet. Ist die Bauteilnummer zu diesem Zeitpunkt im Geschäftsprozess jedoch noch nicht bekannt, muss diese ausgehend vom Bauteilnamen ermittelt werden. Dies kann evtl. durch Nutzung eines automatisch ausführbaren Services erfolgen oder durch einen interaktiven Arbeitsschritt (*Human Task*). In beiden Fällen muss hierfür der Geschäftsprozess umgestaltet werden, was eine fachliche Entscheidung erfordert. Aus diesem Grund muss diese Entscheidung in der Phase des Fachprozess-Designs getroffen werden. Dies erfordert, dass die existierenden Services bekannt sind und zudem eine Vorgabe existiert, diese zu verwenden, sofern wirtschaftlich sinnvoll. Letzteres wird sehr häufig der Fall sein, da die Wiederverwendung eines existierenden Services nicht nur Zeit und Kosten für eine neue Serviceimplementierung erspart, sondern auch Kosten für dessen zukünftige Wartung und Betrieb. Dagegen ist der Nachteil eines ggf. weniger optimalen Geschäftsprozesses (d.h. langsamer, aufwendiger) vom betroffenen Fachbereich abzuwägen.

Geschäftsregeln: *Business Rules* werden bei der Umsetzung von Geschäftsprozessen verwendet, um z.B. Verzweigungsregeln im Kontrollfluss-Ablauf zu definieren. So können etwa Kunden automatisch in die Kategorien Premium-, Standard- und Problemkunde klassifiziert werden, was Auswirkungen auf die Art und Anzahl von Prüfaktivitäten (z.B. eines Leasing-Antrags) haben kann. Durch Verwendung einer *Business Rule Engine* können solche Re-

geln zentral administriert werden, Schwellwerte leicht geändert werden (z.B. Online mittels eines entsprechenden Werkzeugs und ohne erneutes Deployment) und Regeländerungen teilweise sogar direkt durch Fachbereiche erfolgen.

Selbstverständlich ist die Wiederverwendung von *Business Rules* sinnvoll, um Kosten für die Implementierung und Pflege neuer Geschäftsregeln zu vermeiden. Zudem sollten redundante *Business Rules* vermieden werden, um infolge separater Wartung ein unterschiedliches Verhalten verschiedener Geschäftsprozesse zu vermeiden. So sollte ein Kunde nicht nur in einzelnen Prozessen als Premiumkunde behandelt werden. Mittels Look-ahead kann auch für *Business Rules* eine höhere Wiederverwendung erzielt werden. Der Fachprozess-Designer muss existierende *Business Rules* auffinden können, wofür er eine für Fachanwender verständliche Beschreibung ihrer Funktionalität bzw. ihres Verwendungszweckes benötigt. Zudem muss er eine *Business Rule* eindeutig in einem Geschäftsprozessschritt referenzieren können.

3.2 Anforderungen an Look-ahead

Für das Thema Look-ahead ergeben sich folgende generelle Anforderungen:

- Es gibt eine Beschreibungstechnik für existierende Artefakte für jeden Prozessaspekt, so dass diese bei der Prozessimplementierung verwendet werden können.
- Jede Technik zur Beschreibung oder Suche von IT-Artefakten muss für Fachprozess-Designer leicht verständlich und benutzbar sein.
- Der Informationsgehalt einer Beschreibung muss es erlauben, über die Verwendung eines zugehörigen technischen Artefakts zu entscheiden.
- Zu verwendende technische Artefakte müssen im Fachprozessmodell eindeutig referenzierbar sein.

4 Diskussion

Manche Einzelaspekte für Frontloading und Look-ahead lassen sich in heutigen BPM-Werkzeugen realisieren (für Details siehe [12], [29]). Ein Beispiel hierfür sind organisatorische Aspekte, die im Fachprozess dokumentiert werden können, allerdings lediglich unvollständig und nicht ausreichend eindeutig (vgl. Abb. 1). Andere Aspekte, etwa Flexibilitätsmarkierungen, können z.B. durch zusätzliche textuelle Beschreibungen im Fachprozess bereits ausreichend detailliert dokumentiert werden. Wichtig ist dann allerdings eine geeignete SOA-Methodik, die *Frontloading* und *Look-ahead* verbindlich vorschreibt. Im Folgenden betrachten wir dazu existierende SOA-Methodiken und -Vorgehensmodelle.

Service-Oriented Modeling and Architecture (SOMA) [1] adressiert einige der in diesem Beitrag beschriebenen Einzelaspekte, etwa die Modellierung von Flexibilitätsmarkierungen. Diese werden in der *Identification*-Phase durch Angabe von Geschäftsregeln realisiert. Ausnahmesituationen und Datenobjekte werden in der *Specification*-Phase beschrieben. Die Methodik sieht jedoch nicht vor, bereits existierende Services zu finden und diese im Fachprozess zu verwenden. Auch werden keine Beschreibungen von organisatorischen Aspekten oder Bildschirmmasken gefordert.

Auch im Vorgehensmodell *Quasar Enterprise* [13] können Flexibilitätsmarkierungen definiert werden. Die Erstellung entsprechender Geschäftsregeln erfordert jedoch speziell geschulte Mitarbeiter. Die Suche von Services basierend auf fachlichen Kriterien wird nicht unterstützt und ein fachliches Modell ist nur begrenzt vorhanden.

Die Modellierungsmethodik *M3 für SOA* [11] sieht für die verschiedenen Projektphasen unterschiedliche Modelltypen vor. Organisatorische Aspekte können beim Fachprozess-Design (*Initiation*-Phase) modelliert werden, allerdings nicht ausreichend detailliert. Datenobjekte und Geschäftsregeln lassen sich im Fachprozessmodell beschreiben, aber ohne Unterstützung durch das gegebene BPM-Werkzeug. Es ist kein Look-ahead für bereits existierende Datenobjekte oder Services vorgesehen.

AVE (ARIS Value Engineering) [31] ist ein Vorgehensmodell für BPM-Projekte und orientiert sich stark an ARIS. Deshalb ist eine vollständige Modellierung vieler Prozessaspekte nicht möglich. Die Suche nach Services mittels fachlicher Kriterien im Service-Repository wird nur auf sehr einfache Art und Weise unterstützt.

SOA-Method [23] berücksichtigt die Aspekte Service-Suche und Service-Wiederverwendung. Andere Prozessaspekte werden nicht betrachtet.

Weitere Ansätze, wie *OrVia* [32], *Project-Oriented SOA* [30] oder *SUPER* [32] betrachten ebenfalls Vorgehensweisen zur Entwicklung von Prozessapplikationen im SOA-Kontext. Sie unterstützen lediglich kleinere Teile der beschriebenen Frontloading- und Look-ahead-Aspekte.

Keine der untersuchten Ansätze erfüllt alle Anforderungen (vgl. [12]). Manche Ansätze betrachten zwar einzelne Prozessaspekte, diese können jedoch nur abstrakt beschrieben werden. Eine vollständige und dennoch für den Fachprozess-Designer verständliche Technik, wie wir sie in Abschnitt 2.3 für Bearbeiterzuordnungen skizziert haben, wird von keinem Ansatz unterstützt.

5 Zusammenfassung und Ausblick

Frontloading und *Look-ahead* sind für das Fachprozess-Design in einer SOA essentiell, um Implementierungen von Geschäftsprozessen zu erhalten, die tatsächlich die fachlichen Anforderungen erfüllen und dennoch die realen IT-

technischen Gegebenheiten nicht ignorieren. Wir haben analysiert, bei welchen Prozessaspekten dies erforderlich ist. *Frontloading* wurde exemplarisch für das Thema Bearbeiterzuordnungen detailliert.

Die in diesem Beitrag identifizierten Aspekte sind in realen Geschäftsprozess-Implementierungsprojekten nachhaltig zu erproben bzw. verfeinern. Dabei sind auch fortschrittliche Konzepte für die Visualisierung fachlicher und technischer Prozessmodelle vonnöten [3][4]. Um die Fachprozess-Designer nicht mit vielen Neuerungen zu überfordern, sollten für das jeweilige Projekt besonders relevanten Aspekte ausgewählt werden, z.B. basierend auf Erfahrungen aus früheren Projekten. Dieser Beitrag kann hierbei als Inspiration bzw. „Checkliste“ dienen.

Literatur

[1] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, K. Holley: SOMA: A method for developing service-oriented solutions. IBM Systems Journal (2008)

[2] T. Bauer: Stellvertreterregelungen für Task-Bearbeiter in prozessorientierten Applikationen. Datenbank-Spektrum (2009)

[3] R. Bobrik, M. Reichert, T. Bauer: Requirements for the visualization of system-spanning business processes. In: Proc. DEXA'05 Workshops, pp. 948-954 (2005)

[4] R. Bobrik, T. Bauer, M. Reichert: Proviado - personalized and configurable visualizations of business processes. In: Proc. 7th Int'l Conf. on Electronic Commerce and Web Technologies (EC-WEB'06), LNCS 4082, Springer, pp. 61-71 (2006)

[5] L. Bodenstaff, A. Wombacher, M. Reichert, M.C. Jaeger: Monitoring Dependencies for SLAs: The MoDe4SLA Approach. In: IEEE 5th Int'l Conference on Services Computing (SCC’08), IEEE Computer Society Press, pp. 21-29 (2008)

[6] S. Buchwald, T. Bauer, M. Reichert: Bridging the gap between business process models and service composition specifications. In: Lee, Ma, Liu: Service Life Cycle Tools and Technologies: Methods, Trends, and Advances (2012)

[7] S. Buchwald, T. Bauer, M. Reichert: Durchgängige Modellierung von Geschäftsprozessen in einer Service-orientierten Architektur. In: Proc. Modellierung (2010)

[8] S. Buchwald: Erhöhung der Durchgängigkeit und Flexibilität prozessorientierter Applikationen mittels Service-Orientierung. Dissertation, Universität Ulm (2012)

[9] P. Dadam, M. Reichert: The ADEPT project: a decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - Research and Development, 23(2): 81-97, Springer (2009).

[10] P. Dadam, M. Reichert, S. Rinderle-Ma: Prozessmanagementsysteme: Nur ein wenig Flexibilität wird nicht reichen. Informatik-Spektrum, 34(4): 364-376, Springer (2011)

[11] M. Deeg: SOA Fängt weit vor BPEL an – Serviceorientierte Geschäftsprozessmodellierung als Basis für eine SOA. OBJEKTspektrum, Onlineausgabe (2007)

[12] R. Enderle: Frühe fachliche Modellierung ausführungsrelevanter Prozess-Aspekte – Prozessmodellierung in Zeiten von SOA. Diplomarbeit Universität Ulm (2009)

[13] G. Engels, A. Hess, B., O. Juwig, M. Lohmann, J.P. Richter, M. Voss, J. Willkomm: Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten. Dpunkt (2008)

[14] J. Kolb, P. Hübner, M. Reichert: Automatically generating and updating user interface components in process-aware information systems. In: Proc. 20th Int'l Conf on Cooperative Information Systems, LNCS 7565, Springer, pp. 444-454 (2012)
[15] V. Künzle, M. Reichert: Integrating users in object-aware process management systems: issues and challenges. Proc. BPM'09 Workshops, LNBIP 43, pp. 29-41 (2009)
[16] V. Künzle, M. Reichert: Towards Object-aware Process Management Systems: Issues, Challenges, Benefits. In: Proc. 10th Int'l Workshop on Business Process Modeling, Development, and Support (BPMDS'09), LNBIP 29, Springer, pp. 197-210 (2009)
[17] V. Künzle, M. Reichert: PHILharmonicFlows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice, 23(4): 205-244, Wiley (2011)
[18] A. Lanz, B. Weber, B., M. Reichert: Time patterns for process-aware information systems. *Requirements Engineering Journal*, Springer (2013)
[19] M. Lohrmann, M. Reichert: Efficacy-aware Business Process Modeling. In: 20th International Conference on Cooperative Information Systems (2012)
[20] M. Lohrmann, M. Reichert: Understanding Business Process Quality. In: Business Process Management - Theory and Applications, Studies in Computational Intelligence 444, Springer, pp. 41-73 (2013)
[21] C. Li, M. Reichert, A. Wombacher: Mining business process variants: challenges, scenarios, algorithms. Data & Knowledge Eng, 70(5):409-434, Elsevier (2011).
[22] B. Mutschler, M. Reichert, J. Bumiller: Unleashing the effectiveness of process-oriented information systems: problem analysis, critical success factors and implications. IEEE Trans on Sys, Man, and Cyb, 38(3): 280-291, IEEE Comp Soc Press (2008)
[23] M. Reichert, B. Weber: Enabling Flexibility in process-aware information systems – challenges, methods, technologies. Springer (2012)
[24] M. Reichert, P. Dadam, T. Bauer: Dealing with forward and backward jumps in workflow management systems. Int'l J Software and Systems Modeling, 2(1): 37-58 (2003)
[25] S. Rinderle, M. Reichert: On the controlled evolution of access rules in cooperative information systems. Proc CoopIS'05, LNCS 3760, Springer, pp. 238-255 (2005).
[26] S. Rinderle-Ma, M. Reichert: A formal framework for adaptive access control models. Journal on Data Semantics IX, Springer, LNCS 4, pp. 82-112 (2007)
[27] S. Rinderle-Ma, M. Reichert: Comprehensive life cycle support for access rules in information systems: the CEOSIS project. Enterprise Inf Systems, 3(3).219-251 (2009).
[28] P. Offermann: SOAM - Eine Methode zur Konzeption betrieblicher Software mit einer Serviceorientierten Architektur. Wirtschaftsinformatik (2008)
[29] J. Schmitzl: Durchgängige Modellierung von prozessorientierten Anwendungen mit BPMN 2.0. Diplomarbeit Universität Ulm (2010)
[30] L. Shuster: Project-Oriented SOA. SOA Magazin (2008)
[31] Software AG: ARIS Value Engineering. White Paper (2009)
[32] S. Stein, S. Kühne, J. Drawehn, S. Feja, W. Rotzoll: Evaluation of OrViA framework for model-driven SOA implementations: an industrial case study. In: Proc. 6th Int. Conf. on Business Process Management (2008)
[33] B. Weber, S. Sadiq, M. Reichert: Beyond rigidity - dynamic process lifecycle support: a survey on dynamic changes in process-aware information systems. Computer Science - Research & Development, 23(2): 47-65, Springer (2009)
[34] I. Weber, J. Hoffmann, J. Mendling, J. Nitzsche, J.: Towards a Methodology for Semantic Business Process Modeling and Configuration. In: Proc. 2nd Int. Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (2007)