HNU HOCHSCHULE NEU-ULM
UNIVERSITY
OF APPLIED SCIENCES

HNU Working Paper
Nr. 37

Thomas Bauer

Vormodellierte Flexibilität in Prozess-Management-Systemen: Anforderungen, Vorgehensweisen, Lösungsansätze

Pre-Modelled Flexibility in Process Management Systems: Requirements, Procedures, Solution Approaches

02 / 2017

Dr. Thomas Bauer, Professor für Wirtschaftsinformatik mit Schwerpunkt Datenbanken
Hochschule für angewandte Wissenschaften Neu-Ulm
University of Applied Sciences
Wileystraße 1, D-89231 Neu-Ulm

Abstrakt

Bei Process-aware Information Systems (PAIS) ist manchmal ein flexibles Abweichen vom starr modellierten Prozess notwendig. Ansonsten werden die Benutzer zu stark eingeschränkt. Deshalb wurde im Projekt CoPMoF ein Ansatz entwickelt, bei dem zu erwartende, benötigte Flexibilität bereits zur Buildtime einmalig vormodelliert wird. Dies hat gegenüber dynamischen Abweichungen zur Runtime den Vorteil, dass, durch Nutzung der vormodellierten Informationen, bei jeder Abweichung zur Runtime viel weniger Aufwand für die Benutzer entsteht. Außerdem wird die Anwendung der Flexibilität sicherer, z.B. weil hierfür Benutzerrechte definiert werden können. Es werden die entsprechenden Anforderungen dargestellt, wobei insb. darauf eingegangen wird, welche Informationen zur Buildtime vormodelliert werden müssen. Außerdem wird die Notwendigkeit der einzelnen Anforderungen an Praxisbeispielen erläutert.

Freie Schlagwörter:
Process-aware Information System (PAIS), Prozess-Engine, Modellierungszeit, Flexibilität

Abstract

At process-aware information systems (PAIS) sometimes a flexible deviation from the rigidly designed process becomes necessary. Otherwise the users are restricted too much. In the project CoPMoF an approach was developed that allows to define the required flexibility only once already at build-time. Compared to dynamic changes during run-time, this has the advantage that the usage of the pre-defined information reduces the effort for the end users at each deviation at run-time significantly. In addition, applying flexibility becomes saver; e.g., since user rights can be defined. This paper presents the corresponding requirements, with a special focus on the kind of information that shall be pre-defined at build-time. Additionally, the necessity of the requirements is illustrated with examples from practice.

Keywords:
Process-aware Information System (PAIS), Process Engine, Build-time, Flexibility

JEL-Klassifikation: M15

1 Einleitung

Geschäftsprozesse sind seit vielen Jahren ein wichtiges Thema in der wissenschaftlichen Literatur und auch in der Praxis. Sehr häufig werden hierbei ausschließlich die Themengebiete Modellierung, Optimierung und Simulation von Geschäftsprozessen betrachtet. Allerdings ergeben sich auch durch die automatische Steuerung von Geschäftsprozessen durch Prozess-Management-Systeme (PMS) große Vorteile. Dann erhält man ein Process-aware Information System (PAIS) [RW12], bei dem die Einhaltung des vorgegebenen Prozesses durch das PMS sichergestellt wird (Prozesssicherheit). Außerdem werden die Anwender von nicht-produktiven Aufgaben entlastet, wie dem Suchen der richtigen Programmfunktion oder der im aktuellen Geschäftsvorfall benötigten Daten. Dies erfolgt bei einem PAIS automatisch. Allerdings haben PAIS auch Nachteile: So können sich Benutzer durch die aktive Steuerung gegängelt fühlen. Außerdem kann die Einschränkung der möglichen Ausführungsreihenfolgen der Prozessaktivitäten dazu führen, dass in Sonderfällen bestimmte Reihenfolgen nicht realisierbar sind, obwohl hierfür eine betriebliche Notwendigkeit besteht. Dadurch entsteht ein Nachteil für das Unternehmen.

Um solche Nachteile zu vermeiden, muss flexibel vom modellierten Geschäftsprozess abgewichen werden können [DRR11, RDHI09, SMR+07]. Eine spezielle Art von Flexibilität sind Abweichungen, die bereits zur Modellierungszeit (Buildtime) vormodelliert werden, damit sie zur Ausführungszeit (Runtime) der Prozessinstanzen angewandt werden können (Pre-Designed Flexibility [KN06], Flexibility by Design [SMR+07]). In der wissenschaftlichen Literatur findet sich zu diesem Thema jedoch fast ausschließlich diese Kategorisierung. Eine Detaillierung der entsprechenden Anforderungen und von Ansätze zu deren Umsetzung waren bisher kaum Inhalt von Forschungsarbeiten.

Dieser Aspekt ist der Fokus des Projekts CoPMoF (Controlable Pre-Modeled Flexibility). Die Flexibilität in PMS soll vergrößert werden, aber Veränderungen sollen nicht willkürlich (d.h. voll dynamisch) durch die Benutzer festgelegt werden. Stattdessen wird vorhersehbare, möglicherweise zur Runtime benötigte Flexibilität bereits zur Buildtime vormodelliert. Solche Abweichungen können vom Geschäftsprozess-Modellierer (bzgl. ihrer Auswirkungen) gründlich durchdacht und ggf. vom Prozessverantwortlichen genehmigt werden. Dadurch ist die notwendige Prozesssicherheit weiterhin gewährleistet und Abweichungen sind nur im vorgesehenen Umfang und nur durch Benutzer mit entsprechenden Rechten möglich. Der entscheidende Vorteil ist jedoch, dass, für die Auslösung einer Abweichung, für die Benutzer ein sehr viel geringerer Aufwand entsteht, als für eine dynamische Änderung (evtl. wären sie mit deren Definition sogar überfordert). Angenommen, in einem Geschäftsprozess schlägt eine telefonische Rückfrage beim Kunden fehl. Jetzt könnte eine Aktivität „Nachfrage per Post“ dynamisch in den Geschäftsprozess eingefügt werden. Dazu müssten jedoch alle nachfolgend beschriebenen Festlegungen vom Benutzer des PAIS getroffen werden, die besser zur Buildtime vormodelliert werden sollen. Dann wurde nämlich bereits festgelegt, an welcher Stelle im Kontrollfluss diese zusätzliche Aktivität platziert werden soll (d.h. die Vorgänger- und Nachfolgeraktivitäten wurden bereits festgelegt). Auch der Datenfluss wurde bereits definiert, d.h. die Anbindung von Aktivitäten-Input- und -Output-Parametern an die Prozessvariablen. So bezieht z.B. der Input-Parameter Adresse im Feld Strasse seinen Inhalt von der Prozessvariablen

CustomerAddress und dort vom Attribut CustomerStreet. Ebenso wurde bereits eine geeignete Bearbeiterzuordnung vorgegeben, z.B. „Rolle = Kreditbearbeiter und Abteilung = x“, wobei sich x aus der Prozessvariablen CreditApplication und dort dem Attribut ExecutingUnit ergibt.
In der Literatur finden sich primär folgende Ansätze, um die Flexibilität in PMS zu erhöhen:

- Dynamische Änderungen [RD98, RW12] ermöglichen z.B. das Einfügen von, in der Prozessvorlage nicht vorgesehenen Aktivitäten, für eine einzelne Prozessinstanz. Ebenso wird das Löschen oder Verschieben von Aktivitäten ermöglicht. Einen Überblick über relevante Operationen findet sich in [WRR08]. Zur Umsetzung von nicht vorhersehbaren Änderungen stellen diese sicherlich eine unverzichtbare Funktionalität dar. Für vorhersehbare Ausnahmesituationen und Sonderfälle ist dieser Mechanismus jedoch weniger geeignet, weil (wie bereits erläutert) bei jeder Abweichung ein Mehraufwand für die Benutzer entsteht. Hier ist es deutlich sinnvoller, den Aufwand ein einziges Mal bereits zur Buildtime zu betreiben, und die evtl. benötigte Flexibilität vorzumodellieren.
- Schemaevolution [Ri04] beschreibt das Verändern einer Prozessvorlage. Ziel ist hierbei, dass die Veränderungen nicht nur für zukünftig zu startende Prozessinstanzen gültig sind, sondern auch auf bereits laufende Prozessinstanzen angewandt werden. Auch eine solche Funktionalität ist sicherlich notwendig. Zur Lösung der vorliegenden Fragestellung ist sie allerdings nicht geeignet, weil sie keine Flexibilität für einzelner Prozessinstanzen während deren Ausführung ermöglicht.
- Variantenmanagement [RHB15] ermöglicht unterschiedliche Ausführungsreihenfolgen (Varianten) für einen einzigen Geschäftsprozess. Die Entscheidung, welche Variante verwendet werden soll, hängt dabei vom jeweiligen Kontext ab (z.B. Geschäftsbereich, Land). Dieser Mechanismus ist nicht geeignet, um bei einem „plötzlichen Bedarf“ vom vorgegebenen Ablauf flexibel abzuweichen, weil für einen gegebenen Kontext die Ausführungsreihenfolge der Aktivitäten klassisch (d.h. starr) modelliert ist.

Diese Ansätze lösen also nicht die Aufgabenstellung, Flexibilität vorzumodellieren. Auch in der sonstigen wissenschaftlichen Literatur wird bisher lediglich die entsprechende Kategorie von Flexibilität erwähnt (s.o.), diese Kategorie wird jedoch nicht näher untersucht. Damit verbleibt folgende bisher unbeantwortete Forschungsfrage: Welche Szenarien (d.h. Anforderungen) existieren, bei denen es vorteilhaft ist, zur Buildtime ein flexibles Verhalten vorzumodellieren, und welche Informationen müssen hierbei jeweils vorgegeben werden?
Im Projekt CoPMoF soll ein Ansatz mit folgenden Eigenschaften entwickelt werden (dieser Beitrag betrachtet nur einen Teil davon): Die Anforderungen sollen möglichst viele denkbare Szenarien abdecken, wobei eine Vollständigkeit naturgemäß schwer erreichbar ist. Die resultierenden, mit Flexibilität angereicherten, Prozessvorlagen sollen für Modellierer und „normale Anwender“ gut verständlich sein. Dies ist insbesondere bei fachlichen Prozessmodellen erforderlich, aber auch auf der technischen Ebene, z.B. um inhaltliche Fehler in Prozessmodellen überhaupt erkennen zu können. Andererseits muss die Ausführungssemantik der vormodellierten Flexibilität eindeutig sein. Eine gut verständliche, aber vage Modellierungstechnik verbietet sich dadurch. Hauptziel ist jedoch, dass der Aufwand für die Auslösung einer flexiblen Abweichung während der Prozessausführung (Runtime) minimiert wird.
In diesem Beitrag werden zahlreiche unterschiedliche Szenarien für vormodellierte Flexibilität detailliert dargestellt. Dabei wird auf diverse Einzelaspekte und -anforderungen eingegangen,

um die Szenarien möglichst umfassend und verständlich darzustellen. Es wird nicht nur der Kontrollfluss betrachtet, sondern es werden (sofern hierfür relevant) alle Prozessaspekte [Ja97] berücksichtigt. Außerdem wird die Notwendigkeit der einzelnen Anforderungen an Praxisbeispielen erläutert.
Im nachfolgenden Abschnitt werden einige Begriffe eingeführt und die prinzipielle Funktionsweise von PAIS erläutert. Außerdem wird die Problemstellung an einem Anwendungsbeispiel motiviert, das später wieder aufgegriffen wird. In Abschnitt 3 werden die einzelnen Anforderungen erläutert und Praxisbeispiele hierzu dargestellt. Abschnitt 4 stellt verwandte Arbeiten vor und analysiert, inwieweit sich hiermit die Anforderungen umsetzen lassen. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick auf noch ungelöste Fragestellungen.

2 Grundlagen und resultierende Problemstellungen

Im Folgenden wird beschrieben, welche Art vormodellierter Flexibilität heute üblicherweise bereits in den meisten Ansätzen und kommerziellen Systemen zur Prozesssteuerung vorgesehen ist. Anschließend werden einige Beispiele mittels eines etwas umfassenderen Szenarios dargestellt, bei dem die Vormodellierung der erforderlichen Flexibilität heutzutage oft nicht umsetzbar ist.

2.1 Modellierung und Ausführung von Geschäftsprozessen

Process-aware Information Systems (PAIS) bestehen aus einer Buildtime- und einer Runtime-Komponente. Zur Modellierungszeit (Buildtime) werden Prozessvorlagen erstellt, die den später auszuführenden Geschäftsprozess beschreiben. Hierzu wird ein Prozessgraph modelliert, der aus Aktivitäten besteht, deren Ausführungsreihenfolge mittels Kanten und Bedingung festgelegt wird. Dieser Prozessgraph wird zur Ausführungszeit (Runtime) verwendet, um Prozessinstanzen zu erzeugen. Eine Prozess-Engine arbeitet diese ab, indem für jede aktuell anstehende Aktivitäteninstanz (meist nur Aktivität genannt) entsprechende Einträge in den Arbeitslisten der potentiellen Bearbeiter erzeugt werden. Einer dieser Bearbeiter wählt einen solchen Eintrag aus, so dass er in seinem Client-Programm die Aktivität bearbeiten kann. In vielen Anwendungsdomänen besteht die Bearbeitung darin, ein Formular auszufüllen. Die Modellierungskomponente, die Prozess-Engine und der Worklist-Client bilden gemeinsam das Prozess-Management-System (PMS). Solche Systeme sind als kommerzielle (Standard-)Produkte verfügbar.
Der Kontrollfluss ist der am deutlichsten sichtbare Prozessaspekt [Ja97]. Er definiert die Ablaufreihenfolge mittels des Prozessgraphen (vgl. Abbildung 1). Dessen Knoten stellen die Aktivitäten dar, die von Benutzern ausgeführt werden (Human-Task), automatisch durchgeführte Prozessschritte oder ganze Subprozesse repräsentieren. Außerdem enthält der Prozessgraph Gateways (die Rauten), die Verzweigungen (Split) und Zusammenführung (Join) darstellen. In der wissenschaftlichen Literatur findet sich eine Vielzahl an Pattern zur Modellierung des Kontrollflusses [RH06]. In kommerziellen PMS sind für Verzweigungen üblicherweise Split- und Join-Knoten mit XOR- (ein Zweig wird aufgrund einer Bedingung gewählt), OR- (mehrere Zweige) und AND-Semantik (alle Zweige) verfügbar. Außerdem werden Schleifen unterstützt. Sowohl Verzweigung wie auch Schleifen stellen eine einfache Form von vormodellierter

Flexibilität dar, weil sie dazu führen, dass die Menge der ausgeführten Aktivitäten und deren Ausführungsreihenfolge bei jeder Prozessinstanz unterschiedlich sein können.
Einige PMS ermöglichen sogar die Instanziierung einer dynamisch festzulegenden Anzahl von identischen parallelen Zweigen, wobei deren Anzahl zum Startzeitpunkt dieser „Multi-Instanz-Parallelität“ feststehen muss. Dies entspricht dem Kontrollfluss-Pattern „Multiple Instances with a priori Run-Time Knowledge” [RH06]. Dass sich die Anzahl der Zweige nach diesem Startzeitpunkt verändert („without“ in [RH06]) wird von den meisten PMS und deren Prozess-Metamodellen wie z.B. BPMN nicht unterstützt (vgl. [RH06]).
Zur Realisierung des Datenflusses werden häufig Prozessvariablen verwendet. Diese sind mit den Input- und Output-Parametern der Aktivitäten verbunden, so dass ihr Inhalt beim Start der Aktivität an diese übergeben wird und nach deren Ende die Ergebnisdaten in die Prozess-variablen (zurück-)geschrieben werden. Eine Alternative zu Prozessvariablen sind Datenfluss-kanten, welche die Ausgabeparameter von Aktivitäten direkt mit den Eingabeparametern von Nachfolgeaktivitäten verbinden (vgl. ADEPT [RD98]). In beiden Fällen können auch komplexe Datentypen verwendet werden. So werden meist aus elementaren Datentypen zusammen-gesetzte Objekte und Listen (Arrays) unterstützt. Da letztere eine variable Länge haben, können sie die Basis für eine flexible Prozessausführung bilden.
Für den organisatorischen Aspekt gibt es viele Anforderungen [RAHE05]. In kommerziellen PMS können üblicherweise organisatorische Objekte (z.B. Gruppen, Rollen, Abteilungen, etc.) erzeugt und diesen Benutzer zugeordnet werden. In einigen PMS kann hierbei nicht zwischen unterschiedlichen Typen organisatorischer Objekte unterschieden werden. Für jede Aktivität ist eine Bearbeiterzuordnung definiert. Dies ist eine „Formel“ (z.B. Rolle = Software-Entwickler), die organisatorische Objekte verwendet und mit der die Prozess-Engine die potentiellen Bear-beiter dieser Aktivität berechnet. Diese Personen erhalten dann einen entsprechenden Eintrag in ihre Arbeitsliste oder eine eMail, so dass eine dieser Personen die Aktivität zur Bearbeitung auswählen kann. Eine gewisse Flexibilität lässt sich mittels abhängiger Bearbeiterzuordnung modellieren, weil dann die Aktivität nicht stets denselben Personen angeboten wird. So sollen die Rückfragen zu einer Steuererklärung von derjenigen Person beantwortet werden, die die Steuererklärung erstellt hat. Damit ergibt sich eine abhängige Bearbeiterzuordnung „selber Bearbeiter wie Vorgängerakt. X“.
Einige PMS bieten zudem Eskalationsmechanismen an. Wenn die vordefinierte Bearbeitungs-zeit einer Aktivität überschritten wird, wird automatisch z.B. eine eMail an einen Vorgesetzten versandt, oder der Bearbeiter der Aktivität verändert (automatische Delegation). Hierfür ist zu-sätzlich zur der maximalen Bearbeitungsdauer festzulegen, an wen sich die Eskalation richten soll. Außer einer fest definierten Person sollte auch hier die Festlegung einer Personengruppe mittels einer (abhängigen) Bearbeiterzuordnung möglich sein. Allerdings werden Eskalationen in vielen kommerziellen PMS nicht unterstützt, insb. nicht in dem eigentlich notwendigen Funktionsumfang.
Die Ausführung der Aktivitäten erfolgt häufig mittels Formularen. In manchen PMS sind diese sogar ausgehend von den Input-/Output-Parametern der Aktivitäten automatisch generierbar und dann anpassbar [IBM16]. Es ist aber auch möglich, eigene Web-Formulare oder Rich-Client-Anwendungen zu realisieren, die dann über ein API an die Prozess-Engine angebunden werden. Die Ausführung automatisch ablaufender Prozessschritte realisieren viele Prozess-

Engines mittels Web-Service-Aufrufen [IBM16]. Zur Integration von Legacy-Anwendungen werden ggf. Adapter bereitgestellt. Flexibilität ergibt sich hier lediglich aus der Vielzahl der unterstützen Applikationstypen und der Möglichkeit, bei einem Service-Aufruf einen mächtigen Enterprise-Service-Bus (ESB) dazwischen zu schalten.

2.2 Herausforderungen bzgl. vormodellierbarer Flexibilität

Im Folgenden wird an einem Praxisbeispiel der Bedarf an Flexibilität aufgezeigt. Im Fokus hierbei liegen, wie bereits in Abschnitt 1 erläutert, nicht dynamische Änderungen mit denen auf völlig unerwartete Ereignisse reagiert wird. Stattdessen werden Situationen betrachtet, die zwar Ausnahmefälle darstellen, aber vorhersehbar sind. Deshalb kann bereits zur Buildtime ein geeignetes Verhalten vormodelliert werden.

Abbildung 1 zeigt einen vereinfachten Change-Management-Prozess (CMP), wie er zur Beantragung von Produktänderungen in der Automobilindustrie dient (vgl. [BBP09]). Die Notation ist an BPMN 2.0 [FR12] angelehnt. Mit der Akt. A kann ein beliebiger Mitarbeiter des Automobilherstellers eine Änderung eines Bauteils (z.B. Form der Motorhaube) beantragen. Da die Ausführung einer CMP-Instanz einen erheblichen Aufwand verursacht, kann diese in Akt. B bzw. B' von einer Führungskraft gestoppt werden. Akt. C ermittelt automatisch den Verantwortlichen für das zu verändernde Bauteil durch eine Abfrage an das Produktdaten-Management-System (PDM-System). Dieser bewertet in Akt. D Aufwand und Nutzen der Änderung aus Sicht des Entwicklungsbereichs. In Akt. E identifiziert er betroffene Nachbarteile (z.B. Kotflügel, Kühler), die durch eine Formänderung der Motorhaube ebenfalls angepasst werden müssen. Akt. F erfragt die zugehörigen Bauteil-Details und -Verantwortlichen und speichert die Daten in der Liste Nachbarteile (die anderen Prozessvariablen wurden zur Verbesserung der Übersichtlichkeit weggelassen).

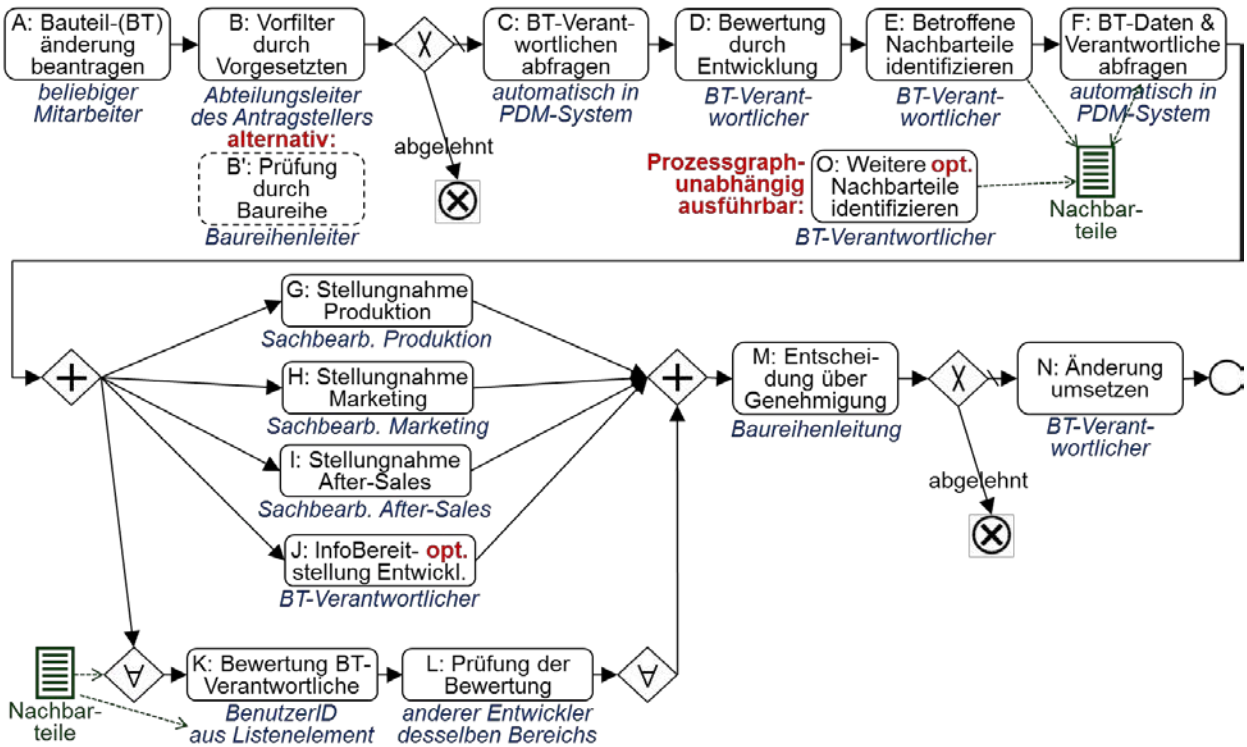

Abbildung 1: Change-Management-Prozess (CMP) für Produktänderungen.

In den Akt. G bis I bewerten diverse Bereiche zeitgleich (parallel), ob die Änderung aus ihrer Sicht realisierbar ist, und welche Kosten bei der Umsetzung entstehen. Die Akt. J erlaubt es dem Bauteilverantwortlichen, hierzu zusätzliche Informationen zur Verfügung zu stellen (die z.B. telefonisch angefragt wurden). Die Bewertung der Änderung aus Sicht der Nachbarteile erfolgt in Akt. K durch den jeweiligen Bauteilverantwortlichen. Diese Aktivität wird mehrfach instanziiert, einmal je Nachbarteil. Dasselbe gilt für Akt. L, in der ein anderer Entwickler die Bewertung überprüft. In Akt. M wird über die Genehmigung des Änderungsantrags entschieden, und ggf. werden die Bauteile in Akt. N geändert.

Die Ausführung des CMP erfordert an einigen Stellen Flexibilität. Einige dieser Stellen sind direkt im Prozessmodell erkennbar, andere werden ausschließlich textuell erläutert:

Die Akt. B' wurde als Alternative zur Akt. B modelliert. B' wird verwendet, falls die Akt. B aktuell nicht ausführbar ist, z.B. weil der disziplinarische Vorgesetzte fachlich für diese Entscheidung nicht ausreichend kompetent ist.

Falls ein Abteilungsleiter seinem Mitarbeiter stark vertraut und weiß, dass eine Änderung sehr eilig ist, dann kann er auch ganz auf die Ausführung der Akt. B verzichten. Durch einen aktiven Eingriff lässt er Akt B (und damit auch B') entfallen (Skip). Diese Möglichkeit ist im Prozessgraphen nicht erkennbar, bei Akt. B muss aber zumindest konfiguriert werden, dass sie ausgelassen werden kann.

Die Akt. J wurde zur Buildtime als optional (opt.) modelliert. Das bedeutet, dass sie entsprechend gekennzeichnet in der Arbeitsliste des Bauteilverantwortlichen erscheint. Dieser kann nun entscheiden, ob Bedarf an einer zusätzlichen Informationsbereitstellung besteht, oder ob er die Aktivität nicht ausführen möchte.

Die Akt. K und L sind in eine Multi-Instanz-Parallelität eingebettet, d.h. die Anzahl Zweige ergibt sich beim ∀-Split aus der Länge der Liste Nachbarteile. Diese Liste wurde von den Akt. E und F befüllt. Sie kann aber auch noch nachträglich durch die Akt. O erweitert werden. Die Akt. O ist unabhängig von Prozessgraphen modelliert. Deshalb ist sie jederzeit ausführbar, macht aber nur bis zum Ende der Multi-Instanz-Parallelität Sinn (d.h. vor dem ∀-Join). Danach können zusätzliche Nachbarteile nicht mehr durch zusätzliche Instanzen der Akt. K und L berücksichtigt werden.

Vor der Genehmigung in Akt. M ist es jederzeit möglich, dass ein fachlicher Fehler entdeckt wird, der bei der Antragstellung oder der Bewertung durch die Entwicklung gemacht wurde. Deshalb ist, ausgehend von allen vor der Akt. M liegenden Aktivitäten, ein Rücksprung zur Akt. A bzw. D erlaubt. Eine Modellierung all dieser Rücksprungkanten würde das Prozessmodell jedoch so unübersichtlich machen, dass hierfür eine andere Lösung benötigt wird.

Dasselbe gilt für Vorwärtssprunge, die jederzeit direkt zur Akt. M auftreten können. Eine solche „Abkürzung“ wird z.B. nötig, falls das Gremium Baureihenleitung eine Änderung unbedingt realisieren möchte (unabhängig von möglichen Nachteilen und Kosten, z.B. wegen gesetzlichen Vorgabe, zur Vermeidung schlechter Presse). In einem solchen Fall sind die Stellungnahmen und Bewertungen (Akt. G bis L) irrelevant.

Der Abbruch einer gesamten Prozessinstanz (Cancel) ist wegen eines entsprechenden Vorstandsbeschlusses jederzeit möglich. Dies kann notwendig werden, falls z.B. spezielle Arten von Änderungen verboten werden, beispielsweise solche, die zu einer Erhöhung des CO_2-

Ausstoßes führen würden. Hierfür muss definiert werden, dass solche Prozessabbrüche erlaubt sind und ggf. auch in welchen Ausführungszuständen einer Prozessinstanz.
Auch beim datenbezogenen Aspekt kann Flexibilität erforderlich sein: Wenn ein Bauteilverantwortlicher während der Ermittlung der Nachbarteile (Akt. E) erkennt, dass er zuvor in Akt. D einen Fehler gemacht hat, so muss er diesen korrigieren können. Hierzu verändert er die Prozessvariable Bewertung, obwohl diese kein Output-Parameter von Akt. E ist, und auch sonst keine Aktivität hierfür vorgesehen ist. Hierzu muss für die Prozessvariable definiert sein, mit welchem Formular bzw. Tool diese Änderung erfolgen kann.
Für die Akt. D und E benötigt der Bauteilverantwortliche einen Stellvertreter. Da es je Bauteil nur einen Verantwortlichen gibt, kann ansonsten der gesamte Ablauf inakzeptabel verzögert werden, falls diese Person z.B. in Urlaub ist. Es genügt jedoch nicht, als Stellvertreter eine (einzige) bestimmte Person festlegen zu können. Zudem ist der Stellvertreter nicht, wie bei vielen anderen seiner Aufgaben, der disziplinarische Vorgesetzter. Dies wäre inadäquat, weil es sich bei Akt. D und E um Projektaufgaben handelt. Deshalb soll abhängig vom betroffenen Fahrzeugprojekt (entsprechend der Prozessvariable Baureihe, z.B. mit Inhalt A-Klasse) ein Kollege bzw. mehrere Kollegen desselben Projekts die Stellvertretung übernehmen.
Die Akt. K der Multi-Instanz-Parallelität hat bei jeder Ausführung einen anderen (potentiellen) Bearbeiter, nämlich den Bauteilverantwortlichen des jeweils zu bewertenden Nachbarteils. Die BenutzerID dieser Person wird in Akt. F vom PDM-System abgefragt und in die Liste Nachbarteile geschrieben. Jeweils ein solcher Listeneintrag wird bei der Ausführung jeder Instanz von Akt. K genutzt, um den hierfür vorgesehenen Bearbeiter zu ermitteln. Solche Abhängigkeiten der Bearbeiterzuordnungen von listenwertigen Prozessvariablen und zudem von der Indexnummer des aktuellen Zweiges müssen zur Buildtime festgelegt werden können, um ein solches Verhalten zu ermöglichen.
Bei der Erstellung der Stellungnahme der Produktion in Akt. G ist Flexibilität bzgl. der hierfür verwendeten Applikation erforderlich: Je nachdem, welcher Sachbearbeiter diese Aktivität ausführt und welche Software auf dessen Computer installiert ist, muss eine unterschiedliche Art von Anwendung verwendet werden. So verfügen einige Benutzer über einen Viewer für CAD-Modelle, andere benutzen den Rich-Client des Produktionsbereichs mit einer speziellen statischen Ansicht für CAD-Modelle, manche ein Web-Formular mit Bauteil-Bildern und wieder andere eine „App“ auf einem Tablet, weil sie häufig in den Produktionshallen unterwegs sind. Bei der Modellierung der Aktivität muss zur Buildtime also festgelegt werden, in welchen Fällen welches Applikationsprogramm gerufen werden soll.

3 Konstrukte zur vormodellierten Flexibilität

Nachfolgend werden Szenarien vorgestellt, bei denen die Flexibilität zur Runtime durch Vormodellierung erhöht wird. Sie werden daraufhin untersucht, was genau zur Buildtime vormodelliert werden muss, um zur Runtime eine große Flexibilität zu ermöglichen, ohne dadurch viel Aufwand für den Benutzer zu verursachen. Hierbei wird zuerst der Aspekt Kontrollfluss betrachtet. Weitere Prozessaspekte (vgl. [Ja97]) werden in den Abschnitten 3.2 bis 3.4 untersucht.

3.1 Kontrollfluss

Einige der dargestellten Szenarien sind ähnlich zu bekannten Kontrollfluss-Pattern [RH06], sie werden nachfolgend jedoch unter dem Gesichtspunkt des Vormodellierens von Flexibilität betrachtet.

KF-1: Optionale Aktivitäten

Bei optionalen Aktivitäten entscheidet der Benutzer, ob eine Aktivität überhaupt ausgeführt werden soll, oder ob sie bei dieser Prozessinstanz irrelevant ist. Hierzu wird die Aktivität in den Arbeitslisten angezeigt, so dass ein Benutzer sie starten kann. Der Arbeitslisteneintrag enthält jedoch zusätzlich eine Kennzeichnung, dass die Aktivität optional ist, und die Möglichkeit sie auszulassen (z.B. mittels eines entsprechenden Buttons bei diesem Eintrag). Bei einer optionalen Aktivität kann es sich auch um eine zusammengesetzte Aktivität handeln, so dass z.B. ein gesamter Subprozess übersprungen wird.
Zur Buildtime muss angegeben werden können (z.B. durch Setzen einer Markierung), dass eine Aktivität optional sein soll (vgl. Akt. J in Abbildung 1). Eine optionale Aktivität kann unter bestimmten Umständen auch hinfällig werden, ohne dass der Benutzer sich aktiv für ein „Auslassen“ entscheidet (s.u. KF-1b und c). Auch dies muss zur Buildtime konfigurierbar sein.
KF-1a: Der Normalfall bei optionalen Aktivitäten ist, dass diese ausgeführt werden. Um sie auszulassen, muss der Benutzer explizit aktiv werden. So wird beim CMP aus Abbildung 1 die Akt. J dem Bauteilverantwortlichen so lange angeboten, bis er sie bearbeitet oder sich für das Auslassen entscheidet. Dies ist sinnvoll, weil die erzeugte Information sowohl von den parallel ausgeführten Akt. G bis I, wie auch von der Nachfolgeakt. M genutzt werden kann.
Befindet sich eine optionale Aktivität innerhalb einer Parallelität, so kann auch ein anderes Verhalten gewählt werden:
KF-1b: Mit der optionalen Akt. C in Abbildung 2 kann die ursprünglich in Akt. A erstellte Bauteilbeschreibung detailliert werden, falls sie sich durch die Ausführung von Akt. B verändert hat. Diese Bauteilbeschreibung wird von der Akt. E in das Stücklistensystem übertragen. Allerdings wartet Akt. E nicht auf die Ausführung oder das Auslassen von Akt. C, weil die Stücklistendaten von anderen Prozessen dringend benötigt werden und Verzögerungen nicht akzeptabel wären. Ist jedoch der untere Zweig beendet (d.h. der AND-Join erreicht), so macht die Ausführung von Akt. C keinen Sinn mehr. Deshalb wird sie von der Prozess-Engine automatisch ausgelassen (sofern dies zur Buildtime so festgelegt wurde).
KF-1c: Das Beispiel aus Abbildung 2 kann auch so abgewandelt werden, dass ein anderes Verhalten erforderlich wird: Befinden sich an der mit * markierten Stelle (d.h. nach Akt. E) weitere Aktivitäten, so ist Akt. C nicht erst dann hinfällig, wenn der untere Zweig den AND-Join erreicht. Stattdessen muss modelliert werden, dass Akt. C automatisch ausgelassen wird, sobald die Akt. E ausführt wurde. Akt. E stellt somit eine Art Milestone dar.

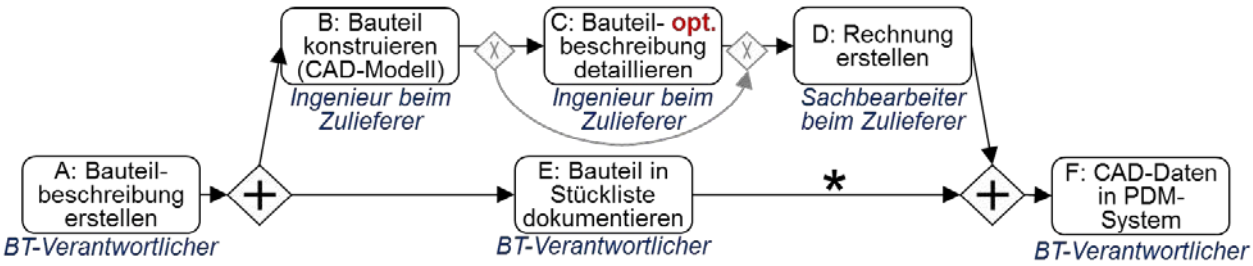

Abbildung 2: Konstruktion eines Bauteils durch einen Zulieferer.

Optionale Aktivitäten lassen sich nicht einfach mittels eines normalen XOR[1] realisieren, so wie grau in Abbildung 2 angedeutet. Abgesehen davon, dass sich KF-1b und c so nicht umsetzen lassen, ergibt sich auch eine andere (bzw. keine) Benutzerinteraktion: Bereits am XOR-Split wird die Verzweigungsentscheidung anhand einer Regel und Werten von Prozessvariablen getroffen. Im Falle einer Auslassung würde die Akt. C also erst gar nicht in die Arbeitslisten der potentiellen Bearbeiter gestellt werden. Im anderen Fall wäre sie nicht mehr auslassbar. Dies entspricht nicht dem bei optionalen Aktivitäten gewünschten Verhalten, so dass es sich hierbei tatsächlich um ein eigenständiges Kontrollfluss-Konstrukt handelt.

KF-2: Alternative Aktivitäten

Die Notwendigkeit für alternative Aktivitäten soll an einem medizinischen Beispiel motiviert werden. Ein Patient erhält normalerweise einen Wirkstoff X in Form von Tabletten. In Sonderfällen erkennt der Arzt, dass der Wirkstoff X bzw. die orale Einnahme ungeeignet ist. Deshalb entscheidet er sich für die alternative Aktivität B und verabreicht dem Wirkstoff Y per Spritze.
Das PMS verhält sich bei alternativen Aktivitäten folgendermaßen: Die Standardaktivität A wird den Benutzern in ihren Arbeitslisten angezeigt. Zusätzlich dargestellt wird ein Hinweis, dass es (eine) alternative Aktivität(en) gibt. Der Benutzer kann sich durch eine aktive Aktion in der Arbeitsliste für eine alternative Aktivität entscheiden (außer bei KF-2d, s.u.).
Zur Buildtime wird festgelegt, welche der folgenden Varianten verwendet werden soll, d.h. wann und wie die Entscheidung für eine alternative Aktivität getroffen wird.
KF-2a: Der Benutzer entscheidet dies vor Reservierung (d.h. dem Start) der Aktivität. Dies ist der Normalfall (und auch der einfachste Fall).
KF-2b: Der Benutzer kann sich auch noch während der laufenden Bearbeitung von Akt. A für Akt. B entscheiden. Dann wird die reguläre Akt. A automatisch abgebrochen und Akt. B gestartet.
KF-2c: Der Benutzer erkennt viel später im Prozess, dass die alternative Aktivität geeigneter gewesen wäre. Die alternative Aktivität wird dann später und zusätzlich ausgeführt. Dies macht z.B. bei Aktivitäten zur Datenerfassung Sinn, wenn durch die alternative Aktivität mehr bzw. andere Daten erfasst werden, die sich im Kontext des aktuellen Geschäftsvorfalles (verspätet) als erforderlich herausgestellt haben. Die zuvor erzeugten Prozessdaten werden also nachträglich verändert.

[1] Eine Realisierung mittels eines XOR-Split mit Deferred-Choice-Semantik wäre denkbar, jedoch wird diese Variante von kommerziellen Prozess-Engines üblicherweise nicht unmittelbar angeboten [RH06].

KF-2d: Die alternative Aktivität wird von der Prozess-Engine automatisch ausgewählt, falls die eigentlich vorgesehene Aktivität fehlschlägt. Hierbei kann es sich um einen fehlgeschlagenen Service-Aufruf bei automatischen Aktivitäten handeln. Aber auch bei manuell bearbeiteten Aktivitäten kann die Prüfung der Nachbedingungen ergeben, dass sie nicht korrekt ausgeführt wurden (z.B. fehlende oder inkonsistente Ergebnisdaten).
Zur Buildtime muss auch festgelegt werden können, wer das Recht hat, sich für die Alternative zu entscheiden. Meist wird ein (potentieller) Bearbeiter dieser Aktivität die Entscheidung treffen dürfen. Es soll aber auch möglich sein, diesen Personenkreis einzuschränken, z.B. auf (besonders kompetente) Personen, die eine zusätzliche Rolle innehaben.
Falls die Aktivität zusammengesetzt ist, entscheidet man sich für die Ausführung eines alternativen Subprozesses, anstatt einer elementaren Aktivität. Dann ist nicht nur der nächste Aktivitätenbearbeiter von dieser Entscheidung betroffen, sondern auch alle anderen Bearbeiter des Subprozesses. Deshalb muss die Entscheidung evtl. durch einen speziellen Verantwortlichen (z.B. Projektleiter) getroffen werden, der selbst nicht zwangsweise ein Bearbeiter der Aktivitäten des Subprozesses ist.
Selbstverständlich können für eine Aktivität auch mehrere alternative Aktivitäten vormodelliert werden, aus denen der Benutzer dann eine geeignete auswählen kann.
Auch alternative Aktivitäten sind kaum mittels eines (normalen) XOR-Splits realisierbar, weil dann die Entscheidung zu früh getroffen werden muss (vgl. KF-1), d.h. bevor die Aktivität in die Arbeitslisten eingestellt wird.

KF-3: Sprünge innerhalb des Prozessgraphen

KF-3a: Sprünge vorwärts

Bei dem in Abbildung 3a dargestellten Reiseantragsprozess erfolgt die Genehmigung der Reise in Akt. G. Ungenehmigte Reisen sind verboten und die Stellungnahmen und Kostenschätzungen (Akt. B bis F) können lange dauern. Bei Reiseanträgen für kurzfristige Termine kann es vorkommen, dass ausnahmsweise, ausgehend von diesen Aktivitäten, direkt zur Akt. G (Genehmigung) gesprungen werden muss, weil sonst ein extrem wichtiger Termin verpasst wird und großer wirtschaftlicher Schaden entsteht.
Im einfachsten Fall findet ein Sprung statt, bevor die Quellaktivität (d.h. Startpunkt des Sprungs) gestartet wurde. Falls die Quellaktivität bereits gestartet wurde, muss sie vor dem Sprung abgebrochen werden. Hat z.B. in Akt. B der Vorgesetzte bereits mit der Erstellung seiner Stellungnahme begonnen und erkennt dann, dass die Reise sehr eilig ist, so wird Akt. B abgebrochen und der Prozess direkt mit Akt. G fortgesetzt. Bei der Modellierung muss also festgelegt werden, ob ausgehend von dieser Quellaktivität ein Sprung erlaubt ist, ob sie hierfür (automatisch) abgebrochen werden darf und welches mögliche Zielaktivitäten des Sprungs sind. Wie in Abbildung 3a grau dargestellt, ist dies hier ausschließlich die Akt. G. Außerdem muss festgelegt werden, wer einen solchen Sprung auslösen darf, z.B. der aktuelle Bearbeiter der Aktivität oder ein fachlicher Prozessverantwortlicher.
Für die übersprungenen Aktivitäten ist festzulegen, ob sie nachgeholt werden sollen. Der Normalfall ist, sie endgültig auszulassen. Es kann jedoch auch erforderlich sein, dass eine übersprungene Aktivität nachgeholt wird. Dies ist zur Buildtime festzulegen, inkl. einer Angabe, ob das Nachholen zu einem beliebigen späteren Zeitpunkt erfolgen darf, oder ob es spätestens

vor dem Start einer bestimmten anderen Aktivität erfolgen muss. Ob im Beispiel von Abbildung 3a eine Bewertung (Akt. B bis F) nachgeholt werden muss, ist abhängig davon, ob die erzeugten Daten im weiteren Ablauf unbedingt benötigt werden. So muss z.B. die Akt. E nachgeholt werden, weil ihre Output-Daten (Höhe der Tagespauschalen) von der Akt. M zur Berechnung der zu erstattenden Reisekosten benötigt werden.

Eine Modellierung der Sprünge mittels Kanten und (normalen) XOR-Splits ist kaum sinnvoll. Wie bereits bei KF-1 diskutiert, werden die Kantenbedingungen des vor Akt. B platzierten Split-Knotens (vgl. Abbildung 3b) bereits direkt nach Beendigung von Akt. A ausgewertet, d.h. es ist kein Sprung mehr möglich, während die Akt. B in den Arbeitslisten auf ihre Ausführung wartet. Hinzu kommt, dass XOR-Splits und entsprechende Kanten vor allen Akt. B bis F eingefügt werden müssen. Wie in Abbildung 3b erkennbar ist, wird der Prozessgraph dadurch sehr unübersichtlich. Dieser Effekt verstärkt sich noch, wenn mehrere Zielaktivitäten erlaubt sind, weil zu all diesen Aktivitäten eine eigene Kante führen muss. Bei Abbildung 3b ist zu beachten, dass es sich um einen sehr kleinen Prozessgraphen handelt. Würde man in Abbildung 1 Sprungkanten ausgehend von den Akt. B bis L zur Akt. M einzeichnen, so wäre die Unübersichtlichkeit noch deutlicher. Deshalb solle ein Modellierungswerkzeug die Sprungkanten ausblenden und vom regulären Kontrollfluss unterscheidbar visualisieren können (z.B. so wie in Abbildung 3a: grau und mehrere Quell-/Zielaktivitäten als gemeinsame Region).

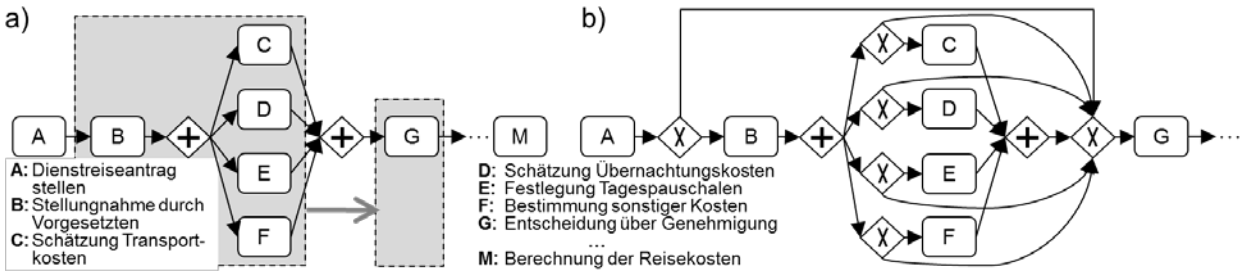

Abbildung 3: Prozess zur Beantragung und Abrechnung einer Dienstreise

KF-3b: Sprünge rückwärts

Angenommen, beim CMP aus Abbildung 1 wird während der Ausführung von Akt. E bis L entdeckt, dass bei Akt. D fehlerhafte Daten eingegeben wurden. Deshalb soll keine Genehmigung (Akt. M) erfolgen, sondern zur Akt. D zurückgesprungen und diese wiederholt werden. Danach wird der Prozess (mit korrekten Daten) ab Aktivität E erneut durchlaufen.

Hier sind mehrere Quellknoten (Akt. E bis L) für den Sprung vorgesehen. Ebenso wie bei Vorwärtssprüngen (KF-3a) sollen diese nicht einzeln mit Kanten und XOR-Splits modelliert werden. Ähnlich wie in Abbildung 3a dargestellt, soll stattdessen wieder eine Quell- und eine Zielregion angegeben werden. Außerdem muss festgelegt werden, wer das Recht hat, einen solchen Sprung auszulösen.

Da nach einem Rücksprung der Prozessgraph wieder vorwärts durchlaufen wird, muss für jede Aktivität festgelegt werden, was dann mit ihren ursprünglichen Ergebnissen (Output-Daten) passieren soll. Hier sind 3 Varianten denkbar und es hängt von der „inhaltlichen Bedeutung“ der Aktivität ab, welche geeignet ist.

1. Verwerfen: Die ursprünglichen Ergebnisse werden verworfen und die Aktivität wird „ganz normal“ erneut ausgeführt
2. Kontrollieren: Die Aktivität wird zwar nochmals ausgeführt, die ursprünglichen Output-Daten

bleiben aber erhalten. Dem Benutzer wird ein vorausgefülltes Formular angezeigt und er soll diese Daten nun kontrollieren und ggf. korrigieren.

3. Beibehalten: Die Aktivität wird nicht erneut ausgeführt und ihre Output-Daten bleiben unverändert erhalten.

Es wird nun am Beispiel des CMP erläutert, warum alle 3 Varianten erforderlich sind: Zum Zeitpunkt des oben erwähnten Rücksprungs von Akt. L zu D seien die Akt. G bis I bereits beendet. Für die Akt. H wurde „Beibehalten“ (3.) festgelegt. Diese Aktivität muss nach dem Rücksprung nicht nochmals ausgeführt werden, weil durch Akt. D geänderte Entwicklungsdetails für das Marketing irrelevant sind. Für Akt. G wird „Kontrollieren“ (2.) festgelegt. Die Ergebnisse dieser Aktivität werden sich selten verändern, müssen jedoch geprüft und ggf. angepasst werden. Eine solche Veränderung ergibt sich aus Sicht der Fahrzeugproduktion, falls durch die Bauteiländerung die Montage schwieriger geworden ist. Für Akt. I wird „Verwerfen“ (1.) verwendet, d.h. die Aktivität muss vollständig neu ausgeführt werden. Ihre ursprünglichen Output-Daten sind irrelevant (werden verworfen), weil ein geändertes Bauteil im Reparaturfall andere Beschaffungs- und Einbaukosten verursacht. Die Varianten 1 und 2 (sofern anwendbar) haben den Vorteil, dass beim erneuten Durchlaufen der Aktivitäten Zeit und Arbeitsaufwand eingespart werden.

In manchen Fällen ist es bei einem Rücksprung erforderlich, dass die ursprünglich ausgeführten Aktivitäten rückgängig gemacht werden. Wird z.B. bei einem Bestellprozess über die Auftragsvergabe an einen Zulieferer zurückgesprungen, so muss diese widerrufen werden oder der Lieferant muss informiert werden, dass er diesen Auftrag ruhen lassen soll. Hierfür muss eine Kompensationsaktivität modelliert werden, die genau diese Aufgabe übernimmt. Hierbei handelt es sich im Prinzip um eine normale Aktivität, die aber speziell für Rücksprungaktionen modelliert wird (also nicht zum regulären Kontrollfluss gehört).

KF-3c: Sprünge bei Parallelitäten

Bei Sprüngen in einen parallelen Bereich hinein oder aus einem heraus ergeben sich einige zusätzliche Fragestellungen. Wie in Abbildung 4 angedeutet, werden die möglichen Quell- und Zielregionen zur Buildtime festlegt (ebenso wie die bereits beschriebenen Berechtigungen etc.). Aufgrund der Parallelität wird hierfür jeweils eine Menge von Aktivitäten angegeben. So sind z.B. in Abbildung 4a für den oberen Zweig die Akt. C und D, für den mittleren die Akt. F und G, sowie für den unteren die Akt. I mögliche Sprungziele. Möchte ein Benutzer nun einen Sprung auslösen, so muss er eine Aktivität je parallelem Zweig als Sprungziel festlegen, sofern hierfür keine Defaults modelliert wurden.

Vorwärtssprung: Bei dem in Abbildung 4a dargestellten Vorwärtssprung werden für 3 Zweige Zielknoten für einen Sprung benötigt. Der Benutzer gibt hierfür z.B. die Akt. C, G und I an. Die Angabe für den unteren Zweig kann auch entfallen, da in diesem Zweig ohnehin nur die Akt. I als Sprungziel in Frage kommt. Um den Aufwand für den Benutzer zu reduzieren, kann bei den anderen Zweigen ein Default-Zielknoten festgelegt werden, z.B. die erste Aktivität nach dem AND-Split (C und F). Dann kann der Benutzer einen Sprung auslösen, bei dem er nur die Akt. G als Ziel angibt. Dadurch wird von der Akt. A zu den Akt. C, G und I gesprungen, so dass diese nun bearbeitbar sind.

Bezüglich dem Nachholen der übersprungenen Aktivitäten gibt es wieder die in KF-3a beschriebenen Varianten.

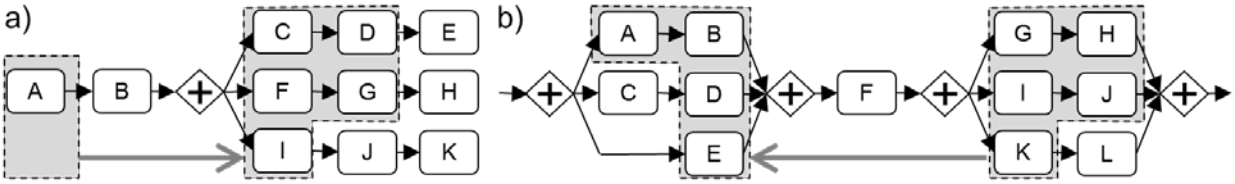

Abbildung 4: Abstraktes Beispiel für Sprünge a) in Parallelität hinein und b) heraus

Rückwärtssprung: Aktuell seien in Abbildung 4b die Akt. G, J und K zu bearbeiten, wobei die Bearbeitung der Akt. J und K bereits (von einem anderen Benutzer) gestartet wurde. Der Bearbeiter von Akt. G löst einen Rücksprung aus (mit seiner Akt. G als Quellknoten). Bereits zur Buildtime ist für jeden parallelen Zweig festzulegen, wie sich dessen Aktivitäten bei einem Rücksprung verhalten sollen. Hierfür gibt es folgende Möglichkeiten:

- Eine aktuell laufende Aktivität des Zweiges wird abgebrochen. So soll z.B. die laufende Akt. J abgebrochen werden, weil ihre Ergebnisse bei der erneuten Ausführung ohnehin verworfen werden (s. KF-3b: Verwerfen). Diese Aktivität fertig zu bearbeiten oder sogar neu zu starten wäre verschwendeter Aufwand.
- Eine aktuell laufende Aktivität in dem Zweig kann zu Ende ausgeführt werden. Es werden aber keine Nachfolgeraktivität mehr gestartet. So kann z.B. bei KF-3b (Kontrollieren) die bereits gestartete Akt. K in Abbildung 4b beendet werden, damit der bisher investierte Arbeitsaufwand nicht verloren geht. Eine Ausführung der Nachfolgeraktivität L ist jedoch nicht erwünscht, weil bei KF-3b deren Ergebnisse evtl. auch komplett verworfen werden.
- Der parallele Zweig kann bis zum AND-Join weiter bearbeitet werden. Wird dies z.B. für den unteren Zweig festgelegt, kann nach Beendigung der Akt. K auch noch die Akt. L ausgeführt werden. Dies kann sinnvoll sein, wenn die Ergebnisdaten ohnehin verwendbar sind (s. KF-3b: Kontrollieren und Beibehalten). Dann steht für die Bearbeitung der Akt. K und L viel Zeit zur Verfügung, bis die Ausführung nach dem Rücksprung wieder bei dem AND-Join ankommt. Dadurch sinkt die Gefahr, dass beim erneuten Durchlaufen Verzögerungen entstehen, weil auf die Beendigung der Akt. L gewartet werden muss.

Für die Zielknoten des Sprungs können zur Buildtime wieder Defaults angegeben werden. Dies kann jede Aktivität eines Zweiges sein. Häufig wird dies die erste Aktivität eines Zweiges (z.B. Akt. A in Abbildung 4b) sein, so dass der gesamte Zweig wiederholt wird. Es muss aber auch festgelegt werden können (vom Benutzer als explizites „Sprungziel“ oder als Default), dass ein Zweig gar nicht erneut ausgeführt werden soll. In Abbildung 4b wäre das z.B. die Position nach der Akt. D für den mittleren Zweig.
Wie bereits bei KF-3b dargestellt, gibt es für die wiederholt auszuführenden Aktivitäten wieder die 3 Varianten, dass die Output-Daten verworfen, kontrolliert oder beibehalten werden sollen.

KF-4: Multi-Instanz-Parallelität

Ein Beispiel hierfür wurde bereits in Abschnitt 2.2 vorgestellt. So soll im CMP aus Abbildung 1 ein Bauteil verändert werden. Da sich dadurch evtl. seine Form, Material oder Härte ändert, kann das benachbarte Bauteile beeinträchtigen. Eine Liste (und damit die Anzahl) dieser Nachbarteile wird in Akt. E festgelegt. Entsprechend oft müssen die Akt. K und L ausgeführt werden, jeweils mit unterschiedlichen Input-Daten und Bearbeitern.

KF-4a: Im (bisher erläuterten) einfachsten Fall ist zur Runtime beim Beginn der Multi-Instanz-Parallelität (d.h. beim ∀-Split) bekannt, wie viele parallele Zweige erzeugt werden müssen (Instanzen der Akt. K und L). Dies entspricht dem Kontrollfluss-Pattern „Multiple Instances with a priori Run-Time Knowledge“ [RH06].
KF-4b: Durch Ausführung der Aktivität O können nachträglich weitere Nachbarteile hinzukommen, so dass zusätzliche parallele Zweige erzeugt werden müssen. Für diese werden also später Instanzen der Akt. K erzeugt. Dies entspricht der Variante des Pattern „without a priori Run-Time Knowledge“ [RH06]. Weitere parallele Zweige können hierbei so lange entstehen, bis alle existierenden Zweige der Multi-Instanz-Parallelität abgeschlossen sind.
KF-4c: Als Erweiterung hiervon kann es auch sinnvoll sein, zur Buildtime ein anderes Kriterium dafür festzulegen, wie lange weitere Zweige erzeugt werden dürfen. Eine mögliche Regel hierfür ist, dass nur solange neue Zweige hinzukommen dürfen, bis ein Milestone in einem der Multi-Instanz-Zweige erreicht ist. So sind z.B. keine neuen Zweige mehr möglich, wenn die erste Bewertung durch einen Bauteilverantwortlichen (Akt. K) abgeschlossen ist. Diese wurde zwar noch nicht von einem Kollegen geprüft (Akt. L), so dass der Zweig nicht vollständig beendet ist. Jedoch hat sich der Bearbeiter von Akt. K auf die ursprüngliche Bauteilliste verlassen, so dass diese nicht mehr nachträglich verändert werden darf. Ein anderer Typ von Regeln verwendet Milestones in einem parallelen Zweig. Wenn z.B. in Akt. G die Produktion basierend auf der aktuellen Teileliste ihre Stellungnahme abgegeben hat, dann darf diese Liste nicht mehr verändert werden, so dass auch keine parallelen Zweige mehr erzeugt werden können.

KF-5: Start-Ende-Abhängigkeiten zwischen Aktivitäten

Die in [RH06] vorgestellten Kontrollfluss-Pattern ermöglichen zahlreiche Ausführungsreihenfolgen von Aktivitäten. Dadurch lassen sich Abläufe so modellieren, dass die Benutzer zur Runtime die Flexibilität haben, die Aktivitäten in der tatsächlich gewünschten Reihenfolge auszuführen. Allerdings berücksichtigen diese Pattern ausschließlich die Reihenfolge vollständig ausgeführter Aktivitäten, z.B. Akt. A muss beendet sein bevor B gestartet werden kann. Dies lässt sich erweitern, indem der Start und das Ende von Aktivitäten getrennt betrachtet werden. Dadurch werden zusätzliche (explizit erlaubte) Ausführungsreihenfolgen möglich und damit wird die Flexibilität vergrößert.
Es gibt 4 Möglichkeiten, hiermit Reihenfolgebeziehungen zu definieren, wobei die erste einer normalen Sequenz entspricht:
KF-5a: Ende von A muss vor Start von B sein: EndBeforeStart
KF-5b: Start von A muss vor Start von B sein: StartBeforeStart
KF-5c: Ende von A muss vor Ende von B sein: EndBeforeEnd
KF-5d: Start von A muss vor Ende von B sein: StartBeforeEnd
Eine Abhängigkeit vom Typ EndBeforeEnd (KF-5c) ist in Abbildung 5a für die Akt. B und C dargestellt: die Akt. B (Fahrzeug reinigen) muss beendet werden, bevor die Akt. C (Fahrzeug an Kunde ausliefern) abgeschlossen wird. Es ist nicht möglich, das Fahrzeug danach noch zu reinigen. Es ist jedoch erlaubt, dass die Akt. B und C überlappend ausgeführt werden, wenn z.B. das Fahrzeug während einer Transportpause gereinigt wird. Damit ergeben sich die in Abbildung 5b darstellten möglichen Ausführungsreihenfolgen für diese beiden Aktivitäten.

Eine Beziehung vom Typ StartBeforeStart ist für folgendes Beispiel erforderlich (vgl. Abbildung 5a): Die Akt. C (Fahrzeugauslieferung, s.o.) muss beginnen bevor die Akt. D (Kunde über anstehende Zustellung informieren) beginnt. Ansonsten, d.h. wenn die Information schon früher an den Kunden gehen würde, wäre die Gefahr einer Fehlinformation zu groß. Vor Beginn der Fahrzeugauslieferung ist die Wahrscheinlichkeit nämlich relativ hoch, dass der Transport doch nicht stattfindet, z.B. weil der LKW nicht verfügbar oder defekt ist.
Auch bei diesen Arten von Abhängigkeiten kann es erforderlich sein, optionale Aktivitäten (vgl. KF-1) zu verwenden. So kann die Aktivität B (Fahrzeugreinigung) auch entfallen, wenn das Fahrzeug bereits sauber ist. Dadurch ist zusätzlich die in Abbildung 5c darstellte Ausführungsreihenfolge erlaubt.
KF-5e (Optionale Abhängigkeiten): Als Erweiterung sollen zudem Start-Ende-Abhängigkeiten verwendbar sein, die eine optionale Ordnung festlegen (Kanten zu Akt. F und G in Abbildung 5a). Eine solche Reihenfolge soll eingehalten werden, muss aber nicht. Die idealerweise als erstes auszuführende Akt. E wird in der Arbeitsliste entsprechend gekennzeichnet, aber auch die optionalen Nachfolger F und G sind dort sichtbar. Der Benutzer kann sich also entscheiden, eine dieser Aktivitäten zuerst zu bearbeiten.
Folgendes Klinikbeispiel erläutert die Notwendigkeit einer optionalen Abhängigkeit: Nach einer bestimmten Diagnose (Akt. A in Abbildung 5a) wird bei einem Patienten üblicherweise (entsprechend dem Klinikstandard) zuerst ein EKG gemacht (Akt. E), danach ein Röntgenbild (Akt. F) und am Schluss wird eventuell die optionale Akt. G (MRT-Aufnahme) ausgeführt. Falls aber eine der Untersuchungseinrichtungen aktuell nicht verfügbar oder überlastet ist, so kann auch von dieser Standardreihenfolge abgewichen werden. Alle 3 Aktivitäten befinden sich in den Arbeitslisten der Benutzer, aber Akt. F und G sind als „eigentlich noch nicht auszuführen“ gekennzeichnet. Ist z.B. das EKG-Gerät aktuell belegt und der Patient wird deshalb direkt zum Röntgen geschickt, so kann der dortige Mitarbeiter die Akt. F problemlos ausführen. Die Akt. E und ggf. auch G finden später statt. Bei dem in Abbildung 5a dargestellten Prozessmodell können die Aktivitäten E bis G auch überlappend ausgeführt werden, was in diesem Fall wohl nicht sinnvoll ist. Dies kann mit einer Mutual-Exclusion (KF-5g) verhindert werden.

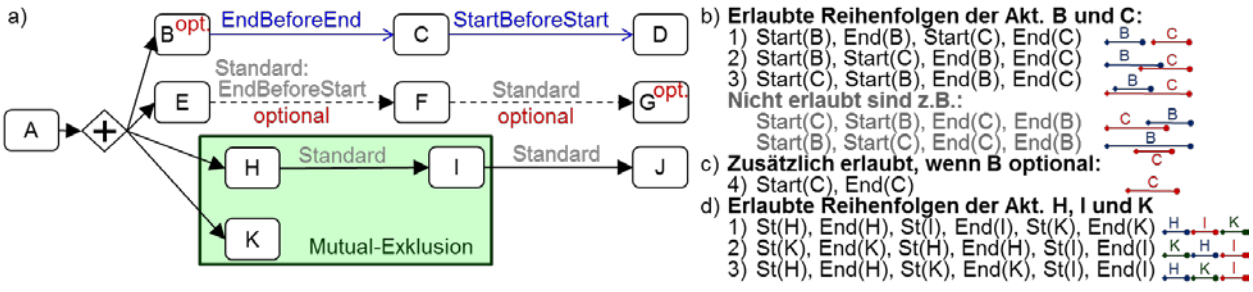

Abbildung 5: Prozess mit Start-Ende-Abhängigkeiten zwischen Aktivitäten

KF-5f (Zeitabstände): Zusätzlich zur Reihenfolgebeziehung kann es erforderlich sein, dass zwischen den Aktivitäten zeitliche Minimal- und Maximalabstände existieren. So muss in Abbildung 5a das in Akt. H gehärtete Bauteil mind. 24 Stunden abkühlen, bevor es lackiert (Akt. I) werden darf. Die Akt. I erscheint deswegen erst dann in den Arbeitslisten der Benutzer, wenn diese 24 Stunden verstrichen sind. Solche zeitlichen Abstände werden ebenfalls im Prozessmodell definiert, sind durch die Prozess-Engine sicherzustellen und können sich auf die Start- und Endezeitpunkte der Aktivitäten beziehen.

KF-5g (Mutual-Exclusion): Bei dem in Abbildung 5a für die Akt. H, I und K dargestellten Konstrukt darf stets nur eine einzige der Aktivitäten bearbeitet werden, die anderen müssen vollständig davor oder vollständig danach bearbeitet werden (vgl. Critical Section und Interleaved Routing in [RH06]). So soll ein zu fertigendes Bauteil zuerst gehärtet (Akt. H) und dann lackiert werden (Akt. I). Danach wird eine Rechnung hierfür erstellt (Akt. J). Parallel dazu wird das Bauteil dem Kunden vorgeführt (Akt. K). Für die Akt. H, I und K ist das Bauteil erforderlich und sie finden an unterschiedlichen Orten statt. Deshalb können sie nicht zeitlich überlappend ausgeführt werden. Dies wird mit einem Mutual-Exclusion-Bereich modelliert. Dieser legt fest, dass keine der enthaltenen Aktivitäten gestartet werden kann, solange eine andere noch läuft. Das führt zu den in Abbildung 5d dargestellten möglichen Ausführungsreihenfolgen. Da sich die Aktivität J nicht im Mutual-Exclusion-Bereich befindet und lediglich nach Akt. I ausgeführt werden muss, ist im Fall 1) eine mit K überlappende Ausführung erlaubt. Bei 3) wird K zwischen H und I ausgeführt. Sollte dies nicht gewünscht sein, so müssten die Akt. H und I als eine einzige zusammengesetzte Aktivität (z.B. Subprozess) modelliert werden.
Die beschrieben Abhängigkeiten (außer dem Normalfall KF-5a) lassen sich mit normalen Kontrollfluss-Beziehungen nicht realisieren: Werden die Aktivitäten bei KF-5b bis d als parallele Aktivitäten modelliert, so sind auch ungewünschte Ausführungsreihenfolgen möglich. Wird eine „normale“ Sequenz (d.h. KF-5a) verwendet, so sind weniger Ausführungsreihenfolgen möglich als gewünscht, d.h. die Flexibilität für die Benutzer wird eingeschränkt. Die Verwendung von optionalen Abhängigkeiten (KF-5e) und Zeitabständen (KF-5f) führt zu einer noch größeren Modellierungsmächtigkeit. Exklusive Abschnitte (KF-5f) lassen sich nur sehr unzureichend mit (normalen) XOR-Splits realisieren. Es müssten alle erlaubten Ausführungsreihenfolgen (vgl. Abbildung 5d) als eigener Zweig modelliert werden, was sehr unübersichtlich wäre (vgl. KF-1). Außerdem würde die Reihenfolge wieder anhand von Prozessdaten und Regeln ermittelt werden, und nicht dadurch, dass Benutzer Aktivitäten zur Bearbeitung auswählen.

KF-6 Prozessgraph-unabhängige Aktivitäten

Es kann Aktivitäten in einem Geschäftsprozess geben, die den Benutzern zusätzlich angeboten werden sollen, d.h. die nicht in den Prozessablauf eingebunden sind. Diese können spontan, aufgrund der Entscheidung eines Benutzers ausgeführt werden. Im CMP aus Abbildung 1 kann jederzeit festgestellt werden, dass in Akt. E ein Nachbarteil vergessen wurde, so dass die Prozessgraph-unabhängige (zusätzliche) Akt. O ausgeführt wird. Diese ergänzt die Liste der Nachbarteile, so dass ein zusätzlicher Zweig der Multi-Instanz-Parallelität gestartet wird (ggf. nachträglich, vgl. KF-4). Eine solche Prozessgraph-unabhängige Aktivität kann vom Benutzer über das Programm-Menü oder mit speziellen Buttons seiner GUI gestartet werden. Eine andere Möglichkeit ist, diese als optionale Aktivität (vgl. KF-1) in der Arbeitsliste des Benutzers darzustellen.
Zur Buildtime müssen Prozessgraph-unabhängige Aktivitäten (wie normale Aktivitäten) vollständig modelliert werden, inkl. einer Bearbeiterzuordnung, der Anbindung ihrer Input-/Output-Parameter an Prozessvariablen (hier: Liste Nachbarteile), etc. Zur Runtime können sie weitgehend wir normale Aktivitäten verwendet werden, so dass für die Benutzer ein viel geringerer Aufwand entsteht (vgl. Abschnitt 1), als für das dynamische Einfügen [RD98] einer zusätzlichen Aktivität.

Für Prozessgraph-unabhängige Aktivitäten muss zudem festgelegt werden, ob die mehrfach ausgeführt werden dürfen. Ggf. ist zusätzlich eine „Prozessregion“ erforderlich, die angibt, wann diese Aktivität ausführbar ist. Beim CMP darf die Akt. O zwar beliebig oft, aber nur vor Ende der Multi-Instanz-Parallelität ausgeführt werden (d.h. vor dem ∀-Join).
Ein Work-around zur Umsetzung dieses Konzeptes wäre die Modellierung eines parallelen Zweiges, der die optionale Akt. O enthält (in einer Schleife, um Akt O mehrfach ausführen zu können). Die Parallelität müsste den gesamten Prozessbereich umschließen, in der die Akt. O gestartet werden kann, also nach Akt. E beginnen und vor dem ∀-Join enden. Werden mehrere Prozessgraph-unabhängige Aktivitäten mit unterschiedlichen Prozessbereichen benötigt, so führt dies zu einem sehr unübersichtlichen Prozessgraphen.

KF-7 Benutzer-Aktionen

Ein Benutzer kann spontane Aktionen durchführen, die den Ausführungszustand einer Prozessinstanz, und damit den Kontrollfluss-Ablauf, verändern. Im Gegensatz z.B. zu optionalen Aktivitäten (KF-1) ist eine solche Aktion selbst nicht vorgeplant, d.h. im Prozessmodell nicht enthalten. Dem Benutzer wird auch nichts Spezielles z.B. in seiner Arbeitsliste dargestellt. Stattdessen werden solche Aktionen mittels des Programmmenüs oder stets vorhandener Buttons ausgelöst. Dennoch sollten bereits zur Buildtime einige Festlegungen für den betroffenen Prozesstyp bzw. die Aktivität getroffen werden, z.B. ob, wann und durch wen solche Aktionen ausgelöst werden dürfen.

KF-7a Abbruch einer Prozessinstanz
Ein Benutzer kann eine gesamt Prozessinstanz abbrechen, die daraufhin sofort beendet wird. Falls dies technisch möglich ist, sollen auch laufende Aktivitäteninstanzen (werden aktuell durch andere Benutzer bearbeitet) abgebrochen oder diese Benutzer informiert werden, dass die Aktivitäten nicht beendet werden müssen. So kann es bei dem CMP aus Abbildung 1 jederzeit, z.B. aufgrund eines Vorstandsbeschlusses, notwendig werden, einen laufenden Änderungsantrag abzubrechen, weil die Änderung ohnehin nicht umgesetzt werden darf.
Zur Buildtime ist für jede Prozessvorlage zu definieren, wann ein Abbruch erlaubt ist. Hier sind folgende Fälle relevant:

1. Ein Abbruch ist für diesen Prozesstyp verboten.
2. Ein Abbruch ist in jedem Ausführungszustand möglich.
3. Ein Abbruch ist nur erlaubt, wenn sich die Ausführung in einer bestimmten Region befindet. Hierzu können z.B. diejenigen Aktivitäten markiert werden, an denen ein Abbruch erlaubt ist.

Im Fall 2 und 3 ist außerdem festzulegen, wer solch einen Abbruch auslösen darf (d.h. Benutzerrechte).

KF-7b Abbruch einer Aktivität
Ein Bearbeiter einer laufenden Aktivität möchte diese abbrechen, z.B. weil er aktuell aus Kapazitätsgründen nicht in der Lage ist, diese zu beenden. Abhängig vom dem Aktivitäten-Life-Cycle der jeweiligen Prozess-Engine muss der Bearbeiter diese Aktivität danach auch noch freigeben (d.h. seine Reservierung aufheben), so dass sie von einem anderen Benutzer reserviert und bearbeitet werden kann. Für Aktivitäten ist hierfür lediglich zu definieren, ob ihr Abbruch erlaubt ist. Es kann auch Aktivitäten geben, die nach erfolgtem Start tatsächlich abge-

schlossen werden müssen, weil bereits unumkehrbare Effekte eingetreten sind (z.B. Abkühlen eines glühenden Bauteils im Wasserbad zur Härtung).

KF-7c Überspringen einer Aktivität (Skip)

Eine Aktivität soll ausgelassen werden, weil sie beim aktuellen Geschäftsvorfall nicht notwendig ist oder nicht ausgeführt werden kann. Diese Benutzeraktion bezieht sich auf nicht optionale Aktivitäten, d.h. auf Aktivitäten, die eigentlich ausgeführt werden müssen. Damit ist ihr Überspringen ein deutlich stärkerer Eingriff, als die Nicht-Ausführung einer optionalen Aktivität (KF-1).

So sei beim CMP aus Abbildung 1 eine Akt. L nicht ausführbar, weil krankheitsbedingt kein anderer Entwickler desselben Geschäftsbereichs für die Überprüfung zur Verfügung steht. Um Verzögerungen zu vermieden, soll deshalb auf diese Überprüfung ganz verzichtet werden, so dass diese Instanz der Akt. L übersprungen wird.

Zur Buildtime ist hierfür für jede Aktivität festzulegen, ob ein Skip überhaupt erlaubt ist. Außerdem muss definiert werden, wer das Überspringen dieser Aktivität auslösen darf.

KF-7d Kompensation einer Aktivität (Undo)

Eine bereits beendete Aktivität soll rückgängig gemacht werden. So kann beim CMP der Bauteilverantwortliche bei der Bearbeitung der Akt. E feststellen, dass er in der bereits abgeschlossenen Vorgängeraktivität D einen Fehler gemacht hat. Also bricht er die Akt. E ab (KF-7b) und macht danach die Akt. D rückgängig, so dass er sie erneut bearbeiten kann.

Zur Buildtime ist für jede Aktivität festzulegen, ob ein Undo überhaupt erlaubt und wer dieses auslösen darf (z.B. der Originalbearbeiter der Aktivität oder ein Prozessverantwortlicher).

Im Normalfall ist ein Undo nur möglich, wenn noch keine Nachfolgeraktivität gestartet bzw. diese abgebrochen wurde (s.o.), oder für diese ebenfalls ein Undo durchgeführt wurde. Es kann jedoch auch ein späteres Undo erlaubt werden. Dann muss ggf. zusätzlich festgelegt werden, bis wann solch ein Undo möglich ist. So kann im CMP eine Stellungnahme (Akt. G bis I) nur widerrufen werden, bevor die darauf basierende Entscheidung über die Genehmigung (Akt. M) erfolgt ist.

Für den Fall eines solchen nachträglichen Undo sollte zudem definiert werden, ob die Aktivität nachzuholen ist, also erneut ausgeführt werden soll, oder ersatzlos entfällt.[2]

Schließlich kann für den Fall eines Undo für jede Aktivität eine Kompensations-Aktivität modelliert werden. Diese hat die Aufgabe, die Auswirkung der ursprünglich ausgeführten Aktivität aufzuheben. So kann die Akt. A „Bauteil aus Lager holen“ durch die Akt. B „Bauteil in Lager bringen“ kompensiert werden. Bei einer späteren erneuten Ausführung von Akt. A kann dann das richtige Bauteil geholt werden.

KF-7e Wiederholung einer Aktivität (Redo)

Eine bereits ausgeführte Aktivität soll erneut durchgeführt werden, z.B. zur Korrektur ihrer Ergebnisdaten. Dies ist z.B. beim CMP der Fall, wenn der Bauteilverantwortliche seine in Akt. K erstellte Bewertung ergänzen oder Fehler beheben möchte. Es kann auch erforderlich sein, in der realen Welt aufgetretene Probleme durch ein Redo zu beheben. Wird in einer Aktivität ein Bauteil aus dem Lager geholt, das sich später als defekt herausstellt, so muss diese

[2] Beim Undo der soeben bearbeiteten Aktivität ist normalerweise immer ein Nachholen (d.h. Wiederholen) nötig, außer die Aktivität wird explizit vom Benutzer mittels Skip (KF-7c) übersprungen.

Aktivität wiederholt werden. Dies sollte nicht nur in der realen Welt, sondern zusätzlich auch im PAIS erfolgen, weil durch das Wiederholen der Aktivität der im IT-System gespeicherte Lagerbestand erneut verkleinert wird und auch die zusätzlich anfallenden Prozesskosten dokumentiert werden.
Zur Buildtime ist für jede Aktivität zu definieren, ob ein Redo überhaupt möglich ist, wer dieses auslösen darf (z.B. der Originalbearbeiter der Aktivität, jeder potentielle Bearbeiter oder ein Prozessverantwortlicher). Außerdem kann eingeschränkt werden, wie lange ein Redo erlaubt ist. So kann im oben genannten CMP-Beispiel die Bewertung der Entwicklung zwar verändert werden (Akt. K), aber dies macht keinen Sinn mehr, nachdem die Akt. M bereits gestartet wurde. Ab diesem Zeitpunkt würden die neu erstellten Ergebnisse bei der Entscheidungsfindung nämlich nicht mehr berücksichtigt werden.

3.2 Datenbezogener Aspekt

Wie in Abschnitt 2 beschrieben, unterstützen PMS üblicherweise Listen mit variabler Elementanzahl, die auch als Basis für eine Multi-Instanz-Parallelität (KF-4) dienen können, um den Zweigen Anwendungsdaten und Bearbeiter zuzuordnen. Darüber hinaus ist die im Folgenden beschriebene Flexibilität erforderlich.

DF-1: Prozessvariablen verändern

In manchen Fällen muss ein Benutzer den Inhalt einer Prozessvariablen ändern können, ohne dass hierfür eine Aktivität im Prozess vorgesehen ist. Dies ist z.B. zur Fehlerkorrektur, für Ergänzungen oder zum nachträglichen Bereitstellen von Daten nützlich.
DF-1a: Zur Buildtime muss festgelegt werden, ob eine bestimmte Prozessvariable überhaupt verändert werden darf und wer dazu berechtigt ist. Außerdem kann der Prozessbereich eingeschränkt werden, in dem eine solche Änderung möglich ist. Wurde beim CMP aus Abbildung 1 in Akt. D ein fehlerhafter Bewertungstext eingegeben und der Prozess ist inzwischen bis zur Akt. F fortgeschritten, so wäre Rücksprung (vgl. KF-3) zur Akt. D und das Wiederholen der Aktivitäten ein unnötiger Aufwand. Einfacher ist eine direkte Datenkorrektur durch den Bauteilverantwortlichen. Diese ist jedoch nur erlaubt, bevor die Stellungnahmen anderer Bereiche (Akt. G bis I) gestartet wurden, weil diese die (falsche) Bewertung als Input verwenden.
DF-1b: Falls die Bewertung aus Akt. D zusätzlich eine Skizze des Bauteils (Powerpoint-Zeichnung oder CAD-Modell) enthält und diese fehlerhaft war, so ist nicht einfach ein Text mit einem Formular o.ä. zu ändern. Stattdessen wird hierfür eine passende Applikation benötigt. Deshalb ist für jede (änderbare) Prozessvariable zu definieren, mit welcher Applikation eine nachträgliche Veränderung erfolgen soll.

DF-2: Zuordnung von Aktivitäten-Parametern zu Prozessvariablen ändern

Auch die Abbildung von Input- und Output-Parametern der Aktivitäten auf die Prozessvariablen muss veränderbar sein. Angenommen, in dem CMP ist aktuell kein Verantwortlicher für das zu ändernde Bauteil definiert (z.B. weil dieser das Unternehmen verlassen hat). Der Bauteilverantwortliche ist aber unbedingt als Input der Akt. D und E notwendig, weil sie sonst keinem

Bearbeiter zugeordnet werden können. Um dieses Problem zu lösen, soll in diesem Fall der Input-Parameter TaskBearbeiterID den Inhalt z.B. von der Prozessvariablen Bereichsleiter beziehen, anstatt von der Prozessvariablen Bauteilverantwortlicher. Hierfür ist zur Buildtime festzulegen, wer eine solche Änderung überhaupt durchführen darf. Zusätzlich kann eingeschränkt werden, welche Prozessvariablen als ein bestimmter Input- bzw. Output-Parameter verwendet werden dürfen.

3.3 Organisatorischer Aspekt

Die potentiellen Bearbeiter einer Aktivität werden mittels einer Bearbeiterzuordnung berechnet. Hierfür sollten ausreichend mächtige Ausdrücke definierbar sein. Außerdem sollte die Prozess-Engine diese Berechnungen regelmäßig erneut ausführen (Refresh), um auch veränderte bzw. neue Zugehörigkeiten von Personen zu Rollen, Gruppen, Abteilungen etc. zu berücksichtigen. Ist bei einem PMS die angebotene Funktionalität unzureichend für die benötigte Art der Bearbeiterberechnung, so könnte folgender Work-around verwendet werden: Die Bearbeiterberechnung wird in eine automatisch durchgeführte Vorgängeraktivität ausgelagert. Diese schreibt die Liste der BenutzerIDs der potentiellen Bearbeiter in eine Prozessvariable, die dann Input-Parameter der eigentlichen Aktivität ist. Nachteilig hierbei ist, dass diese Liste nach Abschluss der Vorgängeraktivität nie mehr aktualisiert wird. Deshalb sollten folgende Anforderungen direkt vom PMS erfüllt werden.

Org-1: Flexible Mechanismen zur Bearbeiterberechnung

Org-1a: Bearbeiter mittels Prozessdaten ermitteln

Es muss möglich sein, in einer Bearbeiterzuordnung Prozessvariablen zu verwenden. Ein Beispiel hierfür ist Akt. D in Abbildung 1 (siehe auch DF-2): Die BenutzerID des Bauteilverantwortlichen wird durch die automatisch durchgeführte Akt. C ermittelt und in einer Prozessvariablen gespeichert. Deren Inhalt wird verwendet, um Akt. D dem richtigen Bearbeiter zuzuordnen. Außer solchen BenutzerIDs können auch andere Daten aus Prozessvariablen für die Bearbeiterberechnung relevant sein: So wird bei der Erstellung eines Kreditantrags die betroffene Bankfiliale in eine Variable FilialID geschrieben. Der Prozessschritt „Kunde über Entscheidung informieren“ soll von einem Sachbearbeiter derselben Filiale durchgeführt werden, so dass sich als Bearbeiterzuordnung ergibt: „Rolle = Sachbearbeiter & OrgEinheit = Wert(FilialID)“.

Bei Multi-Instanz-Parallelitäten (vgl. KF-4) ist es zudem notwendig, die fortlaufende Nummer i des aktuellen Zweiges zu berücksichtigen. So soll jede Bewertungen eines Bauteilverantwortlichen (Akt. K in Abbildung 1) von derjenigen Person durchgeführt werden, die in der Akt. F aus dem PDM-System ermittelt und in der Liste Nachbarteile an der Indexposition i gespeichert wurde (wenn aktuell der i-te Zweig ausgeführt wird).

Die Ermittlung der BenutzerIDs durch die automatischen Akt. C bzw. F hat einen Nachteil: Ändert sich die Verantwortung für ein Bauteil nach Ausführung dieser Abfrage, so wird das nicht mehr berücksichtigt. Deshalb soll ein solcher Service-Aufruf zur Ermittlung einer BenutzerID direkt in die Bearbeiterzuordnung von Akt. D bzw. K integriert werden können. Diese wird dann in regelmäßigen Zeitabständen durch die Prozess-Engine erneut berechnet (Refresh). Dadurch wird ein erneuter Service-Aufruf ausgelöst, so dass sich die tatsächlich aktuell gültigen potentiellen Bearbeiter ergeben.

Org-1b: Alternative Bearbeiterzuordnungen
Es kann vorhersehbar sein, dass die regulären Bearbeiterzuordnungen nicht für alle Sonderfälle geeignet sind. So kann es bei der Gruppe der regulären Bearbeiter zur Überlastungen kommen, die durch Bearbeiter aus anderen Geschäftsbereichen kompensiert werden sollen. Oder es gibt spezielle Prozessinstanzen, die von einem anderen Personenkreis bearbeitet werden sollen (z.B. weil die „normalen Bearbeiter“ hier inhaltlich überfordert wären). Hierfür sollen bereits zur Buildtime alternative Bearbeiterzuordnungen definierbar sein.
Außerdem ist bereits zur Buildtime festzulegen, wer diese Alternative aktivieren darf. Dies können reguläre Bearbeiter von Vorgängeraktivitäten, alle potentiellen Bearbeiter der betroffenen Aktivität oder aber auch Prozessverantwortliche sein. Dieses Umschalten soll auch noch möglich sein, nachdem die Aktivität in Arbeitslisten eingestellt wurde. Dadurch wird zudem automatisch eine Neuberechnung der potentiellen Bearbeiter (Refresh) ausgelöst, so dass die alternative Bearbeiterzuordnung ab diesem Zeitpunkt auch tatsächlich wirksam wird.

Org-2: Stellvertreter

Im CMP dürfen die Akt. D und K ausschließlich von einer einzigen Person, dem Bauteilverantwortlichen, durchgeführt werden. Ist dieser krank, in Urlaub oder auf Dienstreise, so würde das die gesamte Prozessausführung inakzeptabel verzögern. Deshalb muss für diesen Prozessschritt eine Stellvertreter-Regelung (vgl. [Ba09]) festgelegt werden.

Org-2a: Festlegung mittels Regeln
Zur Buildtime werden Regeln festgelegt, mit denen die Stellvertreter berechnet werden. Diese Regeln sollen eine ebenso mächtige Funktionalität ermöglichen, wie normale Bearbeiterzuordnungen. Insbesondere sollen auch organisatorische Objekte (z.B. Rollen, Abteilungen) verwendbar sein, so dass sich bei Änderungen im Organisationmodell automatisch aktualisierte Stellvertreter ergeben. So sollen bei Akt. D und K als Stellvertreter für den Bauteilverantwortlichen alle Personen mit „Rolle = Entwickler und derselben Projektzugehörigkeit wie der reguläre Bearbeiter“ definiert werden können. Wie dieses Beispiel auch zeigt, genügt es nicht, dass eine einzelne Person als Stellvertreter festgelegt werden kann. Stattdessen werden mehrere Stellvertreter benötigt. Diese werden dann ggf. zu potentiellen Bearbeitern, von denen eine Person die Aktivität tatsächlich bearbeiten kann.

Org-2b: Vertretung abhängig vom Aktivitätentyp
Wer die Stellvertreter einer Person sind, soll auch abhängig von der betroffenen Aktivität bzw. ihres Aktivitätentyps festgelegt werden können. So lässt sich der Bauteilverantwortliche durch einen Kollegen desselben Projektes vertreten. Bei Projekt-unabhängigen Aufgaben (z.B. Bestellung von Büromaterial) übernimmt jedoch sein disziplinarischer Vorgesetzter die Stellvertretung. Deshalb sind Stellvertreter-Regelungen im Prozessmodell (zur Buildtime) festzulegen, d.h. für einzelne Aktivitäten(typen) oder die gesamte Prozessvorlage. Es ist nicht ausreichend, vor einer anstehenden Abwesenheit einer Person der Prozess-Engine (Kontext-unabhängig) mitzuteilen, wer dessen Stellvertreter sind.

Org-2c: Konfigurierbarkeit

Es soll konfigurierbar sein (d.h. zur Buildtime festgelegt werden), welches Verhalten des Stellvertretungsmechanismus gewünscht ist. So kann zwischen folgenden Varianten gewählt werden, was jeweils zu einem unterschiedlichen Verhalten führt:

- Es ist festzulegen, wann eine Stellvertretung aktiviert werden soll. Diese kann erfolgen, i) sobald ein einzelner regulären Bearbeiter abwesend ist, ii) wenn alle ihre Stellvertretung aktiviert haben, oder iii) wenn eine bestimme Anzahl oder Quote an Abwesenheiten erreicht wird. Bei dieser Festlegung besteht ein Zielkonflikt zwischen der Vermeidung einer zu hohen Arbeitslast für die verbliebenen regulären Bearbeiter und dem Wunsch, dass diese die eigentlich für sie vorgesehene Aktivität auch tatsächlich bearbeiten.
- Es ist festzulegen, ob ein Stellvertreter selbst wieder vertreten werden kann, die Aktivität also mehrstufig weitergegeben werden soll.
- Es kann gewünscht sein, dass eine Aktivität bei Rückkehr des Originalbearbeiters den Stellvertretern wieder entzogen wird. Dies kann sich ausschließlich auf noch nicht gestartete oder auch auf laufende Aktivitäten beziehen.

Die meisten dieser Anforderungen (für weitere siehe [Ba09]) werden von vielen kommerziellen Prozess-Engines, sofern sie überhaupt Stellvertretungen anbieten, nicht erfüllt.

Org-3: Benutzer-Aktionen

Ähnlich wie beim Kontrollfluss (KF-7) müssen Benutzer auch spontan Aktionen durchführen können, die den organisatorischen Aspekt betreffen. Durch diese ändert sich jedoch nicht der Bearbeitungszustand der Prozessinstanz, sondern die Menge der (potentiellen) Bearbeiter einer Aktivität. Solche Aktionen werden auch in [RAHE05] beschrieben, jedoch wird nachfolgend insbesondere darauf eingegangen, was hierfür zur Buildtime festgelegt werden muss.

Org-3a: Delegation

Bei einer Delegation entscheidet sich der reguläre Bearbeiter einer Aktivität, diese an andere Person(en) weiterzugeben, so dass sie in deren Arbeitsliste erscheint. Hierbei sollen von dem PMS möglichst mächtige Mechanismen unterstützt werden. Es soll eine Delegation an mehrere Personen oder mittels einer Art Bearbeiterzuordnung möglich sein. So möchte z.B. ein Teamleiter eine Aktivität an bestimmte seiner Teammitglieder delegieren, die im gegebenen Kontext besonders geeignet sind. Alternativ kann er die Aktivität auch an alle seine Teammitglieder weitergeben. Dies erfordert besonders wenig Aufwand, wenn hierfür zur Buildtime bereits eine Regel vordefiniert wurde („Rolle = Sachbearbeiter & selbes Team wie der reguläre Bearbeiter“), die direkt zur Delegation verwendet werden kann. Zumindest muss zur Buildtime konfigurierbar sein, ob eine Delegation für eine Aktivität überhaupt erlaubt ist. Sofern dies für die Prozesssicherheit nötig ist (z.B. Einhaltung von Compliance-Regeln), soll der Personenkreis eingeschränkt werden, an den delegiert werden kann. Diese Einschränkung kann wieder durch einen organisatorischen Ausdruck, ähnlich einer Bearbeiterzuordnung, realisiert werden.

Org-3b: Potentielle Bearbeiter verändern

Eine Abwandlung der Delegation ist, dass jemand die Menge der potentiellen Bearbeiter einer Aktivität verändert, indem er Personen hinzufügt oder entfernt. Die anderen potentiellen Bearbeiter (inkl. ggf. er selbst) behalten diese Funktion jedoch. Die Veränderung kann bereits

erfolgen, lange bevor die betroffene Aktivität startbar wird. Hierfür ist zumindest festzulegen, wer eine solche Veränderung vornehmen darf. Wie bei Org-3a kann eingeschränkt werden, welche Personen überhaupt hinzugefügt werden dürfen. Auch diese Funktion ist besonders komfortabel nutzbar, wenn bereits zur Buildtime Regeln vordefiniert werden, mittels derer Personen hinzugefügt oder entfernt werden können. Angenommen, eine Operation kann prinzipiell von jedem Arzt durchgeführt werden. Wenn der Chefarzt nun bei einem vorangehenden Patientengespräch erkennt, dass es sich um einen besonders schwierigen Fall handelt, möchte er z.B. alle Assistenzärzte als Bearbeiter ausschließen. Da solche Fälle öfters vorkommen, wurde hierzu bereits zur Buildtime die Veränderungsformel „entferne Bearbeiter mit Rolle = Assistenzarzt“ vordefiniert, die der hierfür berechtigte Chefarzt nur noch aktivieren muss.

Org-3c: Rückgabe einer Aktivität

Bearbeiter wählen Aktivitäten aus ihrer Arbeitsliste aus. Es kann vorkommen, dass sie erst danach feststellen, dass sie diese Aktivität gar nicht bearbeiten möchten oder können. Dann erfolgt eine Rückgabe. Für jede Aktivität soll konfigurierbar sein, ob eine solche Rückgabe erlaubt ist. Des Weiteren soll festgelegt werden können, (bis) zu welchem Zeitpunkt eine Rückgabe erfolgen kann. Für eine solche Festlegung bieten sich folgende Aktivitätenzustände an:

- Eine Rückgabe ist nur vor Beginn der Bearbeitung dieser Aktivität erlaubt.
- Dies ist zwar nach Bearbeitungsbeginn erlaubt, aber nur solange noch kein Zwischenergebnis erzeugt wurde. Dies ermöglicht das „Anschauen“ der Aktivitäten-Input-Daten (der Details). Es dürfen aber noch keine Output-Daten (zwischen)gespeichert worden sein.
- Eine Rückgabe ist auch nach teilweiser Aktivitätenbearbeitung möglich, d.h. es wurden bereits Zwischenergebnisse erstellt und in Prozessvariablen der Prozess-Engine gespeichert. In diesem Fall kann zusätzlich noch festgelegt werden, ob diese verworfen werden sollen, oder ob sie für den nächsten Bearbeiter den Startzustand der Aktivität darstellen, so dass dieser darauf aufbauend weiterarbeiten kann und keine Arbeit verloren geht.

3.4 Ausführung von Aktivitäteninstanzen

(Kommerzielle) PMS benutzen normalerweise einen Client, der den Benutzern ihre Arbeitsliste dargestellt. Zusätzlich werden von diesem Client Formulare darstellt, mit denen die Aktivitäten bearbeitet werden können. Meist werden hierbei sowohl Web-Clients wie auch Rich-Clients unterstützt. Letztere können auch selbst erstellt und über eine bereitgestellte API an die Prozess-Engine angebunden werden. Darüber hinaus bestehen die nachfolgend beschriebenen Anforderungen, die von heutigen PMS i.d.R. nicht vollständig erfüllt werden.

App-1: Unterschiedliche Applikationstypen

Es sollen die nachfolgend dargestellten Arten von Applikationen verwendbar sein. Der Aufwand für deren Anbindung soll natürlich möglichst gering sein. Hierzu soll das PMS eine entsprechende Anbindung realisieren und anbieten, die zur Buildtime nur noch für die jeweilige Applikation konfiguriert werden muss.

App-1a: Für die Bearbeitung von Aktivitäten darf kein Applikationstyp ausgeschlossen sein, weil die fachliche Notwendigkeit bestehen kann, eine ganz bestimmte Anwendung für eine Aktivität zu verwenden. So kann er erforderlich sein, ein Dokument mit einem Textver-

arbeitungsprogramm (z.B. MS Word) oder einem CAD-Tool zu bearbeiten. Solche „Stand-alone-Applikationen“ lassen sich oft nicht in den PMS-Client integrieren. Dennoch müssen Sie verwendbar sein. Hierbei ist erforderlich, dass sie, nach Auswahl einer Aktivität aus der Arbeitsliste, automatisch gestartet und mit den richtigen Input-Daten versorgt werden. Ebenso muss ihr Ergebnis zur PMS-Engine zurückübertragen werden. Wie dies technisch möglich ist, zeigt z.B. der Program Execution Client von IBM MQ Series Workflow [IBM96].

App-1b: Es müssen auch Funktionen als Applikation verwendbar sein, die aus einem (PMS-externen) Server-System mit eigener Datenverwaltung stammen (z.B. von SAP). In einem solchen Szenario sendet die Prozess-Engine eine Nachricht an den externen Server. Dieser stellt die Aufgabe daraufhin seinen Anwendern zur Verfügung, z.B. indem er ebenfalls Arbeitslisten anbietet oder die Benutzer per eMail benachrichtigt. Eine Integration in die (normalen) Arbeitslisten des PMS ist ebenfalls wünschenswert. Nach Beendigung der Aktivitätenbearbeitung sendet der externe Server die Output-Daten an die Prozess-Engine zurück. Hierbei erhält die PMS-Engine oft nur solche Daten, die zur Ablaufsteuerung erforderlich sind (z.B. für Verzweigungsbedingungen). Die anderen verbleiben ausschließlich beim externen Server-System.

App-1c: Ein PMS muss heutzutage auch Mobile Clients unterstützen. Da sich entsprechende Geräte (im Vergleich zu PCs) stark unterscheiden, sollen auch die Eigenschaften des Mobilgeräts Einfluss auf die Festlegung der potentiellen Aktivitätenbearbeiter haben (vgl. [PRSB16]). So soll deren Auswahl z.B. anhand des Typs des Mobilgeräts (Smartphone, Tablet, Laptop), der Bildschirmgröße, dem Aufenthaltsort oder sogar dem aktuellem Ladezustand getroffen werden. Zur Buildtime muss dann für jede Aktivität konfigurierbar sein, welche Eigenschaften das Mobilgerät haben soll.

App-2: Unterschiedliche Applikationen für eine Aktivität

App-2a: Für eine Aktivität sollen unterschiedliche Applikationen definierbar sein. Diese haben alle dieselbe Schnittstelle (d.h. Input- und Output-Daten), unterschieden sich aber in ihrem Verhalten gegenüber dem Aktivitätenbearbeiter. So kann es erforderlich sein, das Applikationsprogramm flexibel auszuwählen, z.B. abhängig von den Fähigkeiten des Benutzers (Skills), dem vom Benutzer bevorzugten bzw. verwendeten Client-Typ (Web-Client, Rich-Client), der auf dem Computer des Benutzers installierten Software (z.B. MS Word, Open Office Writer) oder dem Gerätetyp (PC, Smartphone, vgl. App-1c). Die Auswahl der Applikation soll mittels Regeln (Formeln, Business Rules) erfolgen, die zur Buildtime definiert werden und Daten zu dem Aktivitätenbearbeiter und Prozessinstanzdaten verwenden.

App-2b: Eine noch weitergehende Anforderung ist, dass auch noch nach dem Deployment der Prozessvorlage weitere Implementierungen einer Aktivität erstellt werden können. Diese sollen natürlich auch verwendet werden, d.h. mittels Regeln nachträglich an die Prozessvorlage angebunden werden (Late Binding). Hierzu muss das Deployment der Aktivitätenimplementierung und der zugehörigen Auswahlregeln unabhängig vom eigentlichen Geschäftsprozess erfolgen. Eine entsprechende Option muss zur Buildtime auswählbar sein

App-2c: Ein Sonderfall einer Aktivität ist eine zusammengesetzte Aktivität (Subprozess, Composed Activity). Auch für solche Aktivitäten soll es möglich sein, diese nicht fest zur Buildtime vorzugeben, sondern mittels Regeln zur Runtime auszuwählen (vgl. App-2a). Es kann

wieder nötig sein, dass nach Deployment des (Vater-)Prozesses und Start der Prozessinstanz, noch weitere Subprozesse erstellt werden. Ebenso wie bei App-2b müssen die Regeln zu deren Auswahl dann nachträglich verändert werden, so dass wieder ein separates Deployment erforderlich wird.

4 Verwandte Ansätze und relevante Literatur

In [KN06] werden verschiedene Arten von Flexibilität für Geschäftsprozesse vorgestellt. Die für unsere Betrachtungen relevante Kategorie ist „Pre-Designed Flexibility“. Die Kategorien werden in [SMR+07, SMR+08] noch verfeinert, so dass die für uns relevanten Kategorien „Flexibility by Design“ und „Flexibility by Underspecification“ entstehen. Auch in [RSS06] werden Kategorien von Änderungen dargestellt, wobei der Fokus auf dynamischen Änderungen und Schemaevolution liegt. Es werden auch „Planned Changes“ erwähnt, die aber nur als „Part of a Process Redesign“ betrachtet werden. Es ist also nicht das Ziel, Flexibilität vorzuplanen und zur Buildtime vorzubereiten. [DRR11] trifft eine Unterscheidung zwischen Flexibilität zur Buildtime und Flexibilität zur Runtime. Ersteres wird allerdings so aufgefasst, dass veränderte Prozessvorlagen schnell zur Ausführung gebracht werden sollen und hierzu zuvor verifiziert und getestet werden müssen. Die Vormodellierung von flexiblen Abläufen wird nicht betrachtet. In all diesen Arbeiten findet sich keine Darstellung einzelner Anforderungen für vormodellierte Flexibilität im Sinne des vorliegenden Beitrags.

Generell finden sich in der Literatur kaum Arbeiten, die speziell die Fragestellung behandeln, was zur Buildtime vormodelliert werden muss, um zur Runtime eine große Flexibilität bei gleichzeitig geringem Aufwand für die Benutzer zu erreichen. Deshalb wird im Folgenden auch auf Arbeiten eingegangen, die ein ähnliches oder weiter gefasstes Themengebiet adressieren. Außerdem werden Ansätze vorgestellt, die zur Umsetzung einzelner Anforderungen aus diesem Beitrag dienen können. Nicht erneut dargestellt werden die in Abschnitt 1 diskutierten Ansätze zu dynamischen Änderungen, Schemaevolution und Variantenmanagement für Geschäftsprozesse, da diese andere Ziele verfolgen.

Der Ansatz, der in der Literatur am häufigsten zur Vormodellierung von Sonderfällen und Ausnahmebehandlungen vorgeschlagen wird, ist ein Exception-Handling auf Basis von Events und speziell modellierter Exception-Handler [LCO+10, RW12]. Für Aktivitäten oder ganze Regionen wird ein (ggf. unterbrechendes) Event definiert. Falls dieses zur Runtime auftritt (Throw), führt es zur Ausführung eines Exception-Handlers (Catch). Dieses Vorgehen ähnelt dem in Programmiersprechen (Try-Catch-Block) und eignet sich insb. zum Abfangen von technischen Fehlern wie z.B. einem Abbruch des Aktivitätenprogramms. Es lässt sich auch auf fachliche Fehler bei der der Prozess-Bearbeitung (z.B. die Modellierung alternativer Aktivitäten, die in Sonderfällen benutzt werden sollten, KF-2) [LCO+10, RW12] oder der Ressourcen-Zuordnung (z.B. Delegation einer Aktivität an einen geeigneteren Bearbeiter, Org-3a) [RAHE05, RW12] übertragen. Allerdings ergibt sich durch die Verwendung von Events und den im Prozessmodell eingebetteten Exception-Handlern ein recht komplexer „Prozessgraph“. Mit diesem dürften die meisten Geschäftsprozess-Modellierer und Fachanwender überfordert sein, da sie normalerweise nicht über den entsprechenden IT-Hintergrund verfügen. Deshalb ist die Verwendung von Events und Exception-Handlern eher ein Work-around, um die in diesem

Beitrag vorgestellten Konzepte auf ein existierendes Prozess-Metamodell (z.B. BPMN) bzw. eine entsprechende Prozess-Engine abzubilden.
In [RDB03] wird beschrieben, wie Sprünge vormodelliert werden, und wie sie zur Runtime auf reguläre Konstrukte des ADEPT-Metamodells abgebildet werden. Damit werden die Anforderungen KF-3a und b (Sprünge vor- und rückwärts, ohne Parallelität) adressiert.
Komplexe Kontrollfluss-Pattern ermöglichen eine gewisse Art von Flexibilität. [RW12] beschreibt in Kap. 4 („Flexibility by Design“) einige recht komplexe Kontrollfluss-Pattern, die viele Ausführungsreihenfolgen und damit vormodellierte Flexibilität ermöglichen. Außer der reinen Modellierung der Geschäftsprozesse werden auch die Themen Ausführungssemantik und Verifikation (d.h. Prüfung der Korrektheit eines Prozessmodells) betrachtet. Jedoch es ist nicht das Ziel, die Anforderungen an Flexibility by Design umfassend darzustellen. Außerdem betreffen diese Pattern ausschließlich den Kontrollfluss und nicht die anderen in diesem Beitrag dargestellten Prozessaspekte. Zwar wird auch auf den Datenfluss eingegangen, jedoch nur im Kontext von Verifikationen und nicht als Aspekt vormodellierter Flexibilität. In [RH06] findet sich eine noch umfassendere Darstellung von Kontrollfluss-Pattern, jedoch liegt auch hier der Fokus nicht auf Anforderungen bzgl. der Vormodellierung von Flexibilität und ausschließlich auf dem Kontrollfluss. In [WRR08] werden sog. „Pattern for Predefined Change“ vorgestellt. Diese ermöglichen, vorzumodellieren, dass bestimmte Entscheidungen oder Festlegungen erst zur Runtime erfolgen sollen. Im Einzelnen sind dies: Late Selection of Process Fragments (vgl. App-2c), Late Modeling of Process Fragments (auch App-2c), Late Composition of Process Fragments und Multi-Instance Activity (KF-4).
In [Kl00] werden für die Prozesse zusätzlich Ziele definiert (Quality of Service Goals, z.B. maximale Durchlaufzeit, Kosten). Wenn diese in Ausnahmefällen verletzt werden, wird vom Standardauflauf abgewichen. Hierzu werden 3 Arten von „flexible Elements“ eingeführt: Alternative Aktivitäten (vgl. KF-2), Non-vital (d.h. optionale) Aktivitäten (KF-1) und Optionale Ausführungsreihenfolgen (diese sollen beachtet werden, aber auch eine parallele Ausführung der Aktivitäten ist erlaubt, das ist ein Teilaspekt von KF-5e). Die tatsächliche Ausführungsreihenfolge ergibt sich dann so, dass die Ziele (trotz Störungen und Ausnahmefällen) möglichst gut eingehalten werden können.
In [RDHI09] wird ein Ansatz vorgestellt, der eine gewisse Art vormodellierter Flexibilität ermöglicht. Hierbei werden Zustände von Daten (Business Objekts) modelliert, wobei durch Signale zusätzlich eine Reihenfolge zwischen diesen festgelegt wird. Die Steuerung der Prozessinstanzen erfolgt durch das Zusammenspiel von verschieden Typen von Business Objekts. Flexibilität wird durch zusätzliche, vom Benutzer optional ausgelöste, Signale realisiert. Diese „Dynamic Signal Types“ realisieren eine vormodellierte Flexibilität für bestimmte Arten von Anforderungen: So kann eine Aufgabe an einen anderen Aktivitätentyp als eigentlich vorgesehen delegiert werden (vgl. KF-2). Außerdem ist die Erzeugung von zusätzlichen Aktivitäteninstanzen oder von zusätzlichen Subprozessen möglich (KF-1). Deren Typ wird vormodelliert und entsprechende Instanzen können nur in unterschiedlichen, ebenfalls vordefinierten, Prozesszuständen (Process Regions) erzeugt werden. Außerdem können „Grenzwerte“ definiert werden, wie oft jede dieser Ausnahmebehandlungen durch die Benutzer ausgelöst werden darf.
Einige Arbeiten beschäftigen sich mit Late-Binding bzw. Late-Selection (vgl. App-2). „Business Rule based Subprocess Selection” [GBS08] ermöglicht, mittels Regeln, zur Runtime einer

Prozessinstanz, einen von mehreren vorgegeben Subprozessen auszuwählen (vgl. App-2c). Enthalten diese unterschiedliche Realisierungen einer Aktivitätenimplementierung, so kann damit auch die Anforderung App-2a umgesetzt werden. In [AHEA06] werden Aktivitäten durch sog. Worklets realisiert. Diese legen abhängig vom Kontext der Prozessinstanz fest, welche der vorgesehenen Aktivitätenimplementierungen verwendet werden soll. Die Worklets selbst sind Subprozesse, die auch noch zur Runtime des eigentlichen Prozesses verändert werden können. Damit wird es möglich, die für einen Prozessschritt zu verwendenden Applikationen und GUIs, jederzeit anzupassen. Dies ist eine ähnliche Vorgehensweise, wie sie auch kommerzielle Prozess-Engines mit Service-Orientierung realisieren (z.B. IBM Business Process Server [IBM16]): Eine Aktivitätenausführung wird durch einen Service-Aufruf umgesetzt, der zum Start eines ESB-Ablaufes (Proxy) führt [Er05, BBP09]. Dieser Ablauf kann Verzweigungen enthalten und damit Kontext-abhängig unterschiedliche Aktivitätenrealisierungen aufrufen. Da das Deployment des ESB-Ablaufes unabhängig von der Prozessvorlage erfolgt, kann er jederzeit ersetzt werden und die Änderung wird auch sofort wirksam. Damit bieten solche Systeme eine geeignet technologische Basis zur Umsetzung der Anforderungen App-2a bis c.

Bei dem Ansatz, der im vorliegenden Beitrag vorgestellt wurde, wird zwar eine große Flexibilität angestrebt, die prinzipiell feste Prozessstruktur soll aber erhalten werden. Die nachfolgend erläuterten Ansätze haben ein anderes Ziel, nämlich einen sehr viel größeren Grad an Flexibilität. Dennoch besteht eine gewisse Ähnlichkeit bei einigen der Einzelanforderungen. Bei [MS02] werden lediglich Prozess-Fragmente (vor-)modelliert, nicht der gesamte Geschäftsprozess. Außerdem wird mittels Constraints (Regeln, Bedingungen) festgelegt, welche dieser Fragmente verwendet werden sollen, welche Abhängigkeiten zwischen ihnen existieren und wann eine Prozessinstanz beendet ist. Basierend darauf kann der Benutzer zur Runtime dynamisch die individuell benötigte Prozessinstanz zusammenbauen. Beim „Case Handling“ [AWG05] handelt es sich um einen Ansatz für wissensintensive Prozesse, bei dem die Daten im Fokus stehen. So kennen die Benutzer (Knowledge Workers) alle Daten und können diese auch spontan ändern (vgl. DF-1). Datenänderungen erfolgen mit Formularen. Der Status der Prozessinstanz ergibt sich aus den Inhalten der Datenobjekte. Diese definieren also, welche Aktivitäten aktuell ausführbar sind, d.h. es wird kein expliziter Kontrollfluss modelliert. Die Benutzer entscheiden selbst, welche dieser Aktivitäten ausgeführt, übersprungen oder wiederholt (vgl. KF-7) werden sollen.

Bei Constraint-basierten Ansätzen wird kein Kontrollflussgraph modelliert. Stattdessen werden Constraints (Regeln) festgelegt, welche die Menge der erlaubten Ausführungsreihenfolgen einschränken. Es sind also alle Ausführungsreihenfolgen erlaubt, die keine der Constraints verletzen. Indem nur wenige Constraints definiert werden, kann eine Vielzahl an erlaubten Ausführungsreihenfolgen modelliert werden. So kann mit geringem Aufwand eine sehr große Flexibilität erreicht werden, d.h. es sind sehr viele Aktivitätenreihenfolgen modellierbar. Einen Überblick über entsprechende Ansätze und deren Funktionsweise bietet [RW12]. Allerdings werden alternative Ausführungspfade bei Constraint-basierten Ansätzen nicht explizit als Sonderfälle modelliert, sind also auch nicht als solche erkennbar. Die Constraints betreffen den Kontrollfluss, d.h. für die anderen Prozessaspekte ist kein flexibles Verhalten gegeben. Außerdem entfällt (meist) die graphische Darstellung der Prozessstruktur, weshalb diese Ansätze für

viele Anwendungsfelder bzw. Geschäftsprozess-Modellierer nicht geeignet sind. So haben Modellierer meist (eingeschränkte) IT-Vorkenntnisse und -Fähigkeiten. Zudem müssen auch die Fachanwender (die evtl. gar keinen Bezug zur IT haben) fähig sein, das resultierende Prozessmodell zu verstehen, zu diskutieren und zu verbessern. Im Folgenden wird auf einige Ansätze eingegangen, bei denen der Aspekt der (vormodellierten) Flexibilität betrachtet wird.
In [BSBB06] wird ein Anwendungsfall aus der Automobil-Industrie dargestellt, bei dem mittels Ziel- und Regel-basierter Modellierung versucht wird, eine größere Flexibilität (Agilität) bei der Prozessausführung zu erreichen. Die Prozesssteuerung erfolgt durch Software-Agenten, die im Falle unerwarteter Ereignisse eine veränderte Ausführungsreihenfolge ermitteln, mit der die Ziele dennoch erreicht werden können. Diese alternativen Ausführungsreihenfolgen sind aber nicht explizit vormodelliert, sondern ergeben sich implizit durch das Zusammenspiel der Ziele und Regeln. [PSSA07] untersucht für Constraint-basierte Prozesse, wie sich dynamische Änderungen und Schemaevolution umsetzen lassen. Hierzu kann der Benutzer, mittels Operationen, Constraints und Aktivitäten hinzufügen oder entfernen. In [RW12] wird zusätzlich eine Operation zur Änderung existierender Constraints betrachtet. [BABM11] ermöglicht, trotz Constraint-basierter Modellierung, die Konsistenz der Regeln zu verifizierten. Außerdem kann aus den Constraints ein Prozessgraph abgeleitet werden, um so die Korrektheit des Prozesses besser beurteilen zu können.

5 Zusammenfassung und Ausblick

In PMS muss vom starr vorgegebenen Prozess abgewichen werden können, damit diese Systeme in der Praxis benutzbar sind. Dies ist mit dynamischen Änderungen möglich, aber bei vorhersehbaren Abweichungen für die Benutzer ein zu großer Aufwand und zudem eine unnötige Fehlerquelle. Deshalb sollten vorhersehbare Sonderfälle und Ausnahmebehandlungen bereits zur Buildtime vormodelliert werden. Solche Szenarien werden in diesem Beitrag vorgestellt, die Anforderungen erläutert und zugehörige Praxisbeispiele dargestellt. Dabei wird nicht nur auf Flexibilität für den Kontrollfluss eingegangen, sondern es werden auch weitere Prozessaspekte betrachtet. Dies alles soll auch einen Beitrag dazu leisten, die Tool-Hersteller zur Unterstützung der beschriebenen Szenarien in kommerziellen PMS bzw. Prozess-Engines zu motivieren.
Die Relevanz und Vollständigkeit der vorgestellten Szenarien sollte noch anhand von Praxisbeispielen, auch aus anderen Domänen, verifiziert werden. Dies wird dadurch erschwert, dass einige der vorgestellten Konstrukte bei einer klassischen Prozessmodellierung nicht vorgesehen sind (z.B. Start-Ende-Abhängigkeiten existieren nicht bei EPKs und BPMN). Deshalb werden entsprechende Situationen in bereits existierenden Prozessmodellen evtl. nicht erfasst worden sein, obwohl sie in der Realität eigentlich existieren.
Einzelne der vorgestellten Anforderungen sollen im Projekt CoPMoF noch weiter detailliert werden. Dies ist z.B. für Sprünge (KF-3) erforderlich. Hier muss die gewünschte Ausführungssemantik noch formal und damit eindeutig definiert werden, weil diese im Zusammenspiel mit Parallelitäten nicht offensichtlich ist. Dasselbe gilt für die Ausführungssemantik bei Start-Ende-Abhängigkeiten (KF-5).

Literaturverzeichnis

[AHEA06] Adams, M.; Hofstede, A. t.; Edmond, D.; Aalst, W. van der: Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In Proc. 14th Int. Conf. on Cooperative Information Systems, 2006; S. 291–308.

[AWG05] Aalst, W. van der; Weske, M.; Grünbauer, D.: Case Handling: A New Paradigm for Business Process Support. In Data & Knowledge Engineering, 2005, 53; S. 129–162.

[Ba09] Bauer, T.: Stellvertreterregelungen für Task-Bearbeiter in prozessorientierten Applikationen. In Datenbank-Spektrum, 2009, 9; S. 40–51.

[BABM11] Boukhebouze, M.; Amghar, Y.; Benharkat, A.; Maamar, Z.: A Rule-based Approach to Model and Verify Flexible Business Processes. In Int. Journal of Business Process Integration and Management, 2011, 5; S. 287–307.

[BBP09] Buchwald, S.; Bauer, T.; Pryss, R.: IT-Infrastrukturen für flexible, service-orientierte Anwendungen - ein Rahmenwerk zur Bewertung. In Proc. 13. GI-Fachtagung Datenbanksysteme in Business, Technologie und Web, 2009; S. 524–543.

[BSBB06] Burmeister, B.; Steiert, H.; Bauer, T.; Baumgärtel, H.: Agile Processes through Goal- and Context-oriented Business Process Modeling. In Proc. Business Process Management Workshops, Workshop on Dynamic Process Management, 2006; S. 217–228.

[DRR11] Dadam, P.; Reichert, M.; Rinderle-Ma, S.: Prozessmanagementsysteme. Nur ein wenig Flexibilität wird nicht reichen. In Informatik-Spektrum, 2011, 34; S. 364–376.

[Er05] Erl, T.: Service-Oriented Architecture - Concepts, Technology, and Design. Prentice Hall, 2005.

[FR12] Freund, J.; Rücker, B.: Praxishandbuch BPMN 2.0. Hanser, 2012.

[GBS08] Graml, T.; Bracht, R.; Spies, M.: Patterns of Business Rules to Enable Agile Business Processes. In Enterprise Information Systems, 2008, 2; S. 385–402.

[IBM16] IBM: IBM Business Process Manager. http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.7, Zugriff am 20.2.2017.

[IBM96] IBM: IBM FlowMark: Installation and Maintenance, 1996.

[Ja97] Jablonski, S.: Architektur von Workflow-Mangement-Systemen. In Informatik Forschung und Entwicklung, Themenheft Workflow-Management, 1997, 12; S. 72–81.

[Kl00] Klingemann, J.: Controlled Flexibility in Workflow Management. In Proc. Int. Conf. on Advanced Information Systems Engineering, 2000; S. 126–141.

[KN06] Kumar, K.; Narasipuram, M.: Defining Requirements for Business Process Flexibility. In Workshop on Business Process Modeling, Design and Support, Proc. of CAiSE06 Workshops, 2006; S. 137–148.

[LCO+10] Lerner, B. S.; Christov, S.; Osterweil, L. J.; Bendraou, R.; Kannengiesser, U.; Wise, A.: Exception Handling Patterns for Process Modeling. In IEEE Transactions on Software Engineering, 2010, 36; S. 162–183.

[MS02] Mangan, P.; Shazia S.: On Building Workflow Models for Flexible Processes. In Australian Computer Science Communications, 2002, 24.

[PRSB16] Pryss, R.; Reichert, M.; Schickler, M.; Bauer, T.: Context-Based Assignment and Execution of Human-Centric Mobile Services. In Proc. IEEE 5th Int. Conf. on Mobile Services, 2016; S. 119–126.

[PSSA07] Pesic, M.; Schonenberg, M.; Sidorova, N.; Aalst, W. van der: Constraint-based Workflow Models: Change Made Easy. In Proc. 15th Int. Conf. on Cooperative Information Systems, 2007; S. 77–94.

[RAHE05] Russell, N.; Aalst, W. van der; Hofstede, A. t.; Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In Proc. Int. Conf. on Advanced Information Systems Engineering, 2005; S. 216–232.

[RD98] Reichert, M.; Dadam, P.: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. In Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, 1998, 10; S. 93–129.

[RDB03] Reichert, M.; Dadam, P.; Bauer, T.: Dealing with Forward and Backward Jumps in Workflow Management Systems. In Software and Systems Modeling, 2003, 2; S. 37–58.

[RDHI09] Redding, G.; Dumas M.; Hofstede, A. t.; Iordachescu, A.: Modelling Flexible Processes with Business Objects. In Proc. IEEE Conf. on Commerce and Enterprise Computing, 2009; S. 41–48.

[RH06] Russell, N.; Hofstede, A. t.: Workflow Control-Flow Patterns. A Revised View, 2006.

[RHB15] Reichert, M.; Hallerbach, A.; Bauer, T.: Lifecycle Management of Business Process Variants. In (J. vom Brocke, M. Rosemann Hrsg.): Handbook on Business Process Management, 2nd Edition. Springer, 2015; S. 251–278.

[Ri04] Rinderle, S.: Schema Evolution in Process Management Systems. Dissertation, Universität Ulm, 2004.

[RSS06] Regev, G.; Soffer, P.; Schmidt, R.: Taxonomy of Flexibility in Business Processes. In Workshop on Business Process Modeling, Design and Support, Proc. of CAiSE06 Workshops, 2006; S. 90–93.

[RW12] Reichert, M.; Weber, B.: Enabling Flexibility in Process-Aware Information Systems. Challenges, Methods, Technologies. Springer, 2012.

[SMR+07] Schonenberg, M.; Mans, R. S.; Russell, N. C.; Mulyar, N. A.; Aalst, W. van der: Towards a Taxonomy of Process Flexibility (Extended Version), Eindhoven University of Technology, 2007.

[SMR+08] Schonenberg, M.; Mans, R. S.; Russell, N. C.; Mulyar, N. A.; Aalst, W. van der: Process Flexibility: A Survey of Contemporary Approaches. In Advances in Enterprise Engineering, 2008, I; S. 16–30.

[WRR08] Weber, B.; Reichert, M.; Rinderle-Ma, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. In Data and Knowledge Engineering, 2008, 66; S. 438–466.