Verhalten und Ausführungssemantik optionaler Kanten in Geschäftsprozessen

Thomas Bauer[1]

Abstract: Für viele Geschäftsprozesse (GP) ist erforderlich, dass das zugrundeliegende Prozess-Management-System (PMS) eine große Flexibilität bei der GP-Ausführung ermöglicht. Ein in der Literatur vorgeschlagener Ansatz, um die Flexibilität zu erhöhen, sind optionale Kanten. Diese geben eine gewünschte Ausführungsreihenfolge von Aktivitäten vor, die von den Benutzern in Ausnahmefällen aber auch ignoriert werden kann. Allerdings ist bisher ungeklärt, welches Verhalten eines PMS und welche Ausführungsreihenfolgen zur Runtime der Prozesse in solchen Fällen exakt gewünscht sind. Dies wird in diesem Beitrag anhand von Prozessbeispielen aus der Praxis geklärt. Außerdem wird eine formale Ausführungssemantik für derartige GP entwickelt. Sie ist erforderlich, damit eine GP-Engine solche Prozesse zur Runtime tatsächlich ausführen kann.

Keywords: Flexibilität, Kontrollfluss, Sequenz, Runtime, Prozess-Engine, Workflow-Engine.

1 Einleitung

Das generelle Ziel des Projektes CoPMoF (Controlable Pre-Modeled Flexibility) ist, die Flexibilität bei der Ausführung von Geschäftsprozessen (GP) zu erhöhen (d.h. zur Runtime), indem vorhersehbar benötigte Funktionalität bereits zu Buildtime „vor-modelliert" wird. Hierfür wurden diverse Ansatzpunkte für den Kontrollfluss [Ba20, Ba21] und weitere Perspektiven [Ba19] von GP skizziert, wie z.B. die Verwendung optionaler Kontrollflusskanten. Solche Kanten geben die gewünschte Ausführungsreihenfolge von Aktivitäten vor; diese muss von den Benutzern aber nicht unbedingt eingehalten werden. Bestimmte, eigentlich noch nicht zur Ausführung anstehende, Aktivitäten werden ihnen deshalb bereits in ihren Arbeitslisten angeboten, allerdings mit einem entsprechenden Warnhinweis. Dadurch sind sie in Ausnahmefällen vorzeitig bearbeitbar. Allerdings wird ein entsprechendes Verhalten in [Ba20, Ba21, Ba23] lediglich angedeutet. Es existieren jedoch noch keine Arbeiten, die das gewünschte exakte Verhalten eines GP mit optionalen Kanten zur Runtime klären. Deshalb wird im Folgenden die offene Forschungsfrage betrachtet, welche genaue Bedeutung optionale Kanten haben sollen und wie die Ausführungsregeln einer GP-Engine hierfür gestaltet sein können.

Optionale Kanten ermöglichen die frühere Bearbeitung von Aktivitäten. Es werden aber weiterhin alle Aktivitäten ausgeführt (d.h. es handelt sich nicht um optionale Aktivitäten [Ba21]). Die parallele Modellierung (d.h. nach einem AND-Split) der mit optionalen Kanten verbundenen Aktivitäten würde zwar alle gewünschten Ausführungsreihenfolgen

[1] Hochschule Neu-Ulm, Fakultät Informationsmanagement, Wileystr. 1, 89231 Neu-Ulm, thomas.bauer@hnu.de, https://orcid.org/0000-0001-8360-8430

ermöglichen, jedoch führt dies zu einem unübersichtlichen GP-Modell und die „bevorzugte“ Ausführungsreihenfolge wäre für die Endbenutzer nicht mehr erkennbar.

Ziel dieses Beitrags ist, ein sinnvolles Verhalten und eine formale Ausführungssemantik für solche GP zu entwickeln. Nach einer Literaturdiskussion (Abschnitt 2) werden hierzu in Abschnitt 3 relevante Beispielprozesse und Szenarien vorgestellt. Mittels dieser wird untersucht, welche Art des Verhaltens zur Runtime sinnvoll ist. In Abschnitt 4 wird hierfür eine formale (operationale) Ausführungssemantik definiert. Eine solche ist unbedingt erforderlich, damit Prozess-Engines entsprechende GP zur Runtime steuern können. Die resultierende Ausführungssemantik ist ähnlich zu der von BPMN 2.0 [OMG11], jedoch werden zusätzliche Ausführungszustände für Aktivitäteninstanzen und zusätzliche bzw. veränderte Ausführungsregeln benötigt. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick.

2 Stand der Forschung

Beim Case Handling [AWG05], dem CMMN-Standard und Constraint-basierten Ansätzen [APS09, Pe07, RW12] wird kein Kontrollflussgraph vorgegeben. So wird sogar eine noch größere Flexibilität als mit dynamischen Änderungen [RD98] ermöglicht, weil die Benutzer auszuführende Aktivitäten recht frei wählen können. Unser Ziel ist hingegen eine höhere Prozesssicherheit (und damit weniger Flexibilität), indem erlaubte Abweichungen vor-modelliert werden (und damit geprüft und genehmigt werden können).

Außer den bereits angesprochenen eigenen Vorarbeiten des Projekts CoPMoF enthält die wissenschaftliche Literatur kaum Erwähnungen optionaler Sequenzkanten. Lediglich im Projekt CrossFlow [Gr00, Kl00] wird ein entsprechender Kantentyp vorgeschlagen und als Optional Execution Order bezeichnet: Um in Ausnahmesituationen z.B. eine vorgegebene Ausführungsdauer erreichen zu können, wird bei diesem Kantentyp von der eigentlich vorgegebenen Reihenfolge abgewichen. Durch eine parallele Ausführung von Aktivitäten kann dadurch die gewünschte maximale Prozessdurchlaufzeit eingehalten werden. In den Kontrollfluss-Pattern [RH06], dem BPMN-Standard [OMG11], sowie in sonstigen auf GP-Graphen basierenden Ansätzen, werden optionale Sequenzkanten nicht betrachtet. Deswegen wird im Folgenden auf optionale Reihenfolgebeziehungen in Ansätzen eingegangen, die keine Kanten und GP-Graphen im klassischen Sinn verwenden.

Bei Constraint-basierte Ansätzen (vgl. [APS09, Pe07, RW12]) werden die erlaubten Ausführungsreihenfolgen mittels Constraints (Regeln) festgelegt. Durch die Verwendung eines optionalen Constraints (Recommendation) kann das Verhalten einer optionalen Sequenzkante erreicht werden. Jedoch erfolgt die Steuerung einer Prozessinstanz, d.h. die Berechnung der aktuell ausführbaren Aktivitäten, unmittelbar durch die Constraints. Deshalb ist bei solchen Ansätzen keine separate Ausführungssemantik erforderlich. Dementsprechend wird in Arbeiten zu optionalen Constraints nichts mit der nachfolgend

vorgestellten Ausführungssemantik Vergleichbares entwickelt. Ebenso wird nicht diskutiert, welche Arten des Verhaltens in komplexeren Szenarien erforderlich sein können (vgl. Abschnitt 3.1 sowie den in Abschnitt 3.2 eingeführten Typ „Soft“).

Beim Case Handling [AWG05] ergibt sich der Status einer Prozessinstanz aus den Inhalten von Datenobjekten, d.h. diese definieren, welche Aktivitäten aktuell ausführbar sind. Bei DOPA [Li06] handelt es sich um solch einen datengetriebenen Ansatz, bei dem durch zusätzliche Regeln sichergestellt werden kann, dass nur tatsächlich erwünschte Ausführungsreihenfolgen möglich sind. Die Regeln definieren also einen Teil des Kontrollflusses. Da eine solche Regel bei DOPA optional sein kann, entspricht dies einer optionalen Sequenzkante. Weil sich die tatsächliche Ausführungsreihenfolge ausschließlich aus Dateninhalten und diesen Regeln ergibt, ist auch bei solchen Ansätzen eine zusätzliche formale Ausführungssemantik nicht erforderlich

Zusammenfassend lässt sich feststellen, dass keine Arbeiten existieren, welche die gewünschten Arten des Verhaltens optionaler Kanten diskutieren oder eine formale Ausführungssemantik hierfür definieren. Damit GP-Modelle mit solchen Kantentypen von einer GP-Engine tatsächlich gesteuert werden können, ist es erforderlich, diese Forschungslücke zu schließen. Dies schafft zudem eine Grundlage, um GP-Modellierungssprachen wie BPMN [OMG11] um optionale Kanten erweitern zu können.

3 Nutzen und Verhalten optionaler Kanten

Im Folgenden wird der Verwendungszweck von optionalen Sequenzkanten erläutert. Außerdem wird geklärt, welches Verhalten für solche Kanten zur Ausführungszeit gewünscht ist. Diese Klärung bildet die Basis für die Entwicklung einer Ausführungssemantik in Abschnitt 4.

3.1 Sequenzen optionaler Kanten

Folgendes Klinikbeispiel dient dazu, den Nutzen optionaler Kanten zu veranschaulichen: Bei bestimmten Diagnosen (Akt. a in Abb. 1a) wird bei einem Patienten üblicherweise (entsprechend dem Klinikstandard) zuerst ein EKG erstellt (Akt. b), danach ein Röntgenbild (Akt. c) und dann wird die Akt. d (MRT-Aufnahme) ausgeführt. Nach Abschluss dieser Untersuchungen wird dem Patienten eine Behandlung vorgeschlagen (Akt. e). Falls aber eine der Untersuchungseinrichtungen aktuell nicht verfügbar oder überlastet ist, kann von dieser Standardreihenfolge abgewichen werden: Nach Beendigung der Akt. a befinden sich die Untersuchungsaktivitäten b bis d alle in den Arbeitslisten der Benutzer. Jedoch ist ausschließlich die Akt. b „regulär startbar“ (d.h. ohne auszunutzen, dass eine Kante optional ist). Die auf Akt. b mit optionalen Kanten folgenden Aktivitäten c und d sind z.B. für den Patienten Fritz Müller in den Arbeitslisten von Peter Meier und von Anne Schmidt (Abb. 1b und c) aber bereits enthalten, jedoch als „eigentlich noch nicht auszuführen“ gekennzeichnet. Sind z.B. aktuell alle EKG-Geräte belegt und der Patient wird

deshalb direkt zum MRT geschickt, so kann die dortige Mitarbeiterin Anne Schmidt die Akt. d bereits problemlos ausführen.[2] Die Akt. b und c finden dann später statt. Wird zuerst die Akt. b ausgeführt, so verschwindet der Warnhinweis aus der Arbeitsliste von Peter Meier (Abb. 1b), weil die Akt. c nun regulär bearbeitbar ist.[3]

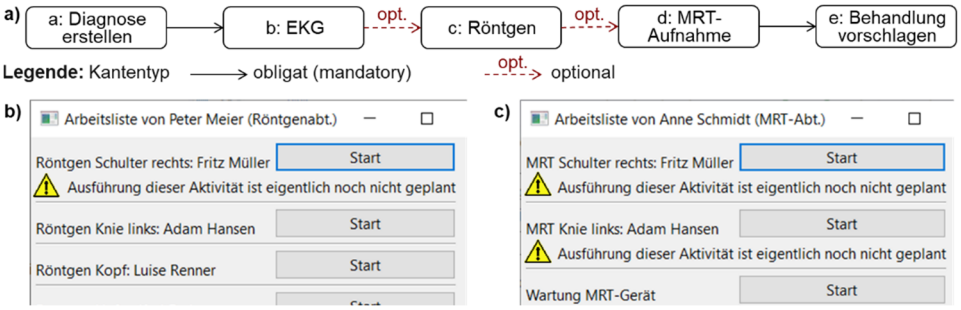

Abb. 1: a) Untersuchungsprozess sowie b) Arbeitsliste für die Akt. c und c) für Akt. d

Die zur Akt. e „Behandlung vorschlagen“ führende Kante ist nicht optional (sondern obligat, engl.: mandatory). Deswegen darf die Akt. e wirklich erst nach Beendigung der Akt. d ausgeführt werden. Allerdings bleibt folgende Frage zu klären: Falls die Akt. d vorzeitig gestartet und auch bereits beendet wurde, muss Akt. e dann dennoch auf die Beendigung der Aktivitäten b und c warten? Dies ist in diesem Beispiel sinnvoll, weil eine konkrete Behandlung erst dann vorgeschlagen werden sollte, wenn alle Untersuchungen durchgeführt wurden. Das bedeutet, die Kante von Akt. d zu e mit dem „Strictness-Typ“ mandatory „synchronisiert“ die Ausführung aller vorherigen Aktivitäten, so dass ab Akt. e nur noch eine einzige Aktivität dieser „Sequenz“ bearbeitbar ist, sich also in den Arbeitslisten befindet oder aktuell ausgeführt wird (so wie bei klassischen Sequenzen). Wie nachfolgend dargestellt, gibt es aber auch Szenarien, in denen ein anderes Verhalten wünschenswert ist.

3.2 Strictness-Typ „Soft“

Bei dem in Abb. 2 dargestellten Ausschnitt eines GP zum Testen eines Bauteils ist solch ein abweichendes Verhalten gewünscht. Nach dem (nicht dargestellten) Bestellen eines Bauteils bei einem Zulieferer und dem Wareneingang, soll das eingetroffene Bauteil zuerst im Testing-IT-System verbucht werden (Akt. a). Danach wird es in Akt. b für den Test

[2] Beim vorliegenden Prozessmodell können die Aktivitäten b bis d prinzipiell auch überlappend ausgeführt werden. Dies wird aber wohl nicht passieren, weil der Patient gleichzeitig nur an einer Untersuchungseinrichtung anwesend sein kann. Deshalb werden für ihn nicht mehrere Aktivitäten (gleichzeitig) gestartet werden, bevor die anderen beendet sind. In anderen Szenarien kann eine solche parallele Bearbeitung jedoch erwünscht sein, um in Ausnahmesituationen Zeit zu sparen (z.B. die Aktivitäten a und b in Abb. 2).

[3] Damit Arbeitslisten mit solcher Funktionalität realisiert werden können, muss über die Schnittstelle (API) der GP-Engine abfragbar sein, ob eine bestimmte Aktivität regulär oder vorzeitig bearbeitbar ist. Dies ist einfach realisierbar, weil dies direkt am Aktivitätenzustand (vgl. Abschnitt 4.1) Active (regulär) bzw. PreActive (vorzeitig) erkennbar ist.

vorbereitet (es werden z.B. Mess-Sensoren angebracht). Der eigentliche Test (Belastbarkeit, Korrosionsbeständigkeit, o.ä.) findet in Akt. c statt, bevor das Testergebnis im bereits erwähnten Testing-IT-System dokumentiert wird.

Um Entwicklungszeit einzusparen, kann mit der Bauteilvorbereitung begonnen werden, bevor die Verbuchung abgeschlossen wurde oder sogar bevor sie gestartet wurde. Deshalb befindet sich zwischen den Akt. a und b eine optionale Kante. Auch mit dem Test (Akt. c) kann vor Abschluss der Akt. a begonnen werden. Eine optionale Kante zwischen den Akt. b und c wäre jedoch nicht sinnvoll, weil mit dem Testen erst begonnen werden kann, wenn die Bauteil-Vorbereitung abgeschlossen ist. Eine „echte" Mandatory-Kante hätte hier jedoch den Nachteil, dass die Akt. c dann auf die Beendigung der Akt. a warten müsste (so wie Akt. e auf die Untersuchungen in Abb. 1a). Dies ist jedoch nicht erforderlich und hierdurch würde unnötig Zeit verschwendet werden. Deshalb wird zwischen den Akt. b und c eine Kante mit dem Strictness-Typ soft verwendet.[4] Dieser stellt einen „Mittelweg" zwischen optional und mandatory mit folgendem Verhalten dar: Die Akt. c muss auf die Beendigung der unmittelbaren Vorgängerakt. b warten (der Startaktivität dieser Soft-Kante). Akt. c muss aber nicht auf die Beendigung von deren mit optionalen Kanten verbundenen Vorgängeraktivitäten (hier nur Akt. a) warten.

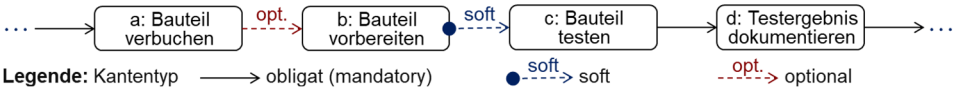

Abb. 2: GP zum Testen eines Bauteil mit dem zusätzlichen Strictness-Typ „Soft"

Die Dokumentation des Testergebnisses in Akt. d kann erst erfolgen, wenn dieses Bauteil durch Akt. a im Testing-IT-System erfasst wurde. Deshalb wird zwischen den Akt. c und d eine Kante mit Strictness-Typ mandatory verwendet. Dies führt (wie bereits bei Abb. 1a erläutert) dazu, dass vor dem Start der Akt. d alle Vorgängeraktivitäten beendet sein müssen.

Die Verwendung einer Kante mit Strictness-Typ soft ist nur nach optionalen Kanten und anderen Soft-Kanten relevant. Bei einer Kante mit Strictness-Typ mandatory müssen ohnehin alle Vorgängeraktivitäten beendet sein, so dass das Verhalten einer danach folgenden Soft-Kante identisch zu einer (klassischen) Mandatory-Kante wäre.[5]

---

[4] Der ausgefüllte Kreis am Kantenbeginn deutet an, dass auf die Beendigung dieser Aktivität gewartet werden muss. Dass die Kante gestrichelt ist, symbolisiert ihre „Optionalität", d.h. auf die Vorgängeraktivitäten von Akt. b muss nicht gewartet werden. Allerdings sind alle verwendeten Symbole nur ein erster Vorschlag und kein Forschungsergebnis dieses Beitrags.

[5] Dies kann von einem GP-Modellierungswerkzeug ausgenutzt werden, indem dieser Strictness-Typ nur nach Aktivitäten mit solchen eingehenden Kanten angeboten wird. Dadurch muss der GP-Designer diesen Typ bei anderen Aktivitäten gar nicht berücksichtigen. Bei Aktivitäten, bei denen Soft-Kanten relevant sind, kann das Tool zudem erklären, welche Auswirkungen die Verwendung eines solchen Typs hat. Dies kann am konkret vorliegenden GP erfolgen. Im Beispiel aus Abb. 2 wird dann die Information angezeigt, dass Akt. c zwar auf die Beendigung von Akt. b warten muss, aber nicht auf Akt. a. Ein solches Verhalten vereinfacht die Verwendung der neuen Kantentypen.

Im Beispiel aus Abb. 2 ist der Zweck der Soft-Kante, dass Akt. c auf die Beendigung der Akt. b warten muss, aber nicht auf deren Vorgängerakt. a. Es ist auch modellierbar, dass auf die Beendigung mehrerer (aber wieder nicht aller) Vorgängeraktivitäten gewartet werden muss: Befindet sich vor einer Akt. c eine Sequenz von mit optionalen Kanten verbundenen Aktivitäten $a_1 \ldots a_m$, $b_1 \ldots b_n$ (in beliebiger Reihenfolge) und es soll nur auf die Beendigung der Aktivitäten $b_1 \ldots b_n$ gewartet werden, dann kann dies wie in Abb. 3 dargestellt modelliert werden: Ausgehend von diesen Aktivitäten $b_1 \ldots b_n$ werden Soft-Kanten modelliert, die alle in die Akt. c münden. Da alle ihre Vorgängeraktivitäten (irgendwann) tatsächlich ausgeführt werden, muss diese Akt. c eine „AND-Join-Semantik" haben. Bei diese Modellierungsweise wird zwar auf die Aktivitäten $b_1 \ldots b_n$ gewartet, nicht aber auf die Aktivitäten $a_1 \ldots a_m$, von denen ausgehend keine Soft-Kante zur Akt. c führt.

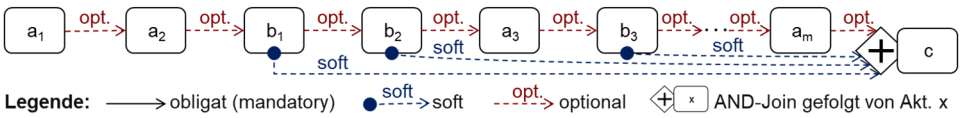

Abb. 3: Sequenz von Aktivitäten, bei denen Akt. c nur auf die Aktivitäten b_i warten muss

3.3 Kombination optionaler Kanten mit Gateways

Gateways (vgl. BPMN [OMG11]) dienen dazu, einen Pfad in mehrere aufzuspalten (Split) oder diese wieder zusammenzuführen (Join). Hierfür existieren einige grundlegende Gateway-Typen, die sich dadurch unterscheiden, dass genau ein Pfad (XOR), mehrere Pfade (OR) oder alle Pfade (AND) ausgeführt werden. Bei den ersten beiden Typen wird mittels Bedingungen festgelegt, welche Pfade ausgeführt werden sollen. Das ist für den Pfad i der Fall, wenn die zur ersten Kante dieses Pfades gehörende Bedingung (condition) $cond_i$ zum Wert true evaluiert.

Das in Abb. 4 dargestellte Beispiel enthält einen OR-Split, so dass abhängig von den Bedingungen $cond_1$ bis $cond_5$ manche der Pfade mit den Aktivitäten b_i bis d_i ausgeführt werden. Dies erfolgt parallel, d.h. zeitlich überlappend. Betrachten wir nun das Verhalten am OR-Join: Hier wird in klassischen GP-Metamodellen auf die Beendigung aller ausgeführter Pfade gewartet. Da der Prozess aus Abb. 4 optionale Kanten enthält, soll jedoch nicht auf die Beendigung aller Pfade gewartet werden, bevor der OR-Join und damit auch Akt. e ausgeführt werden können. Stattdessen ist folgendes Verhalten gewünscht:

- Die Aktivitäten d1 und d2 müssen beendet sein, bevor der OR-Join und Akt. e ausgeführt werden, weil die Kanten zwischen diesen Aktivitäten und dem Join den Strictness-Typ mandatory haben.
  Da in die Akt. d2 eine optionale Kante mündet, ist anzumerken, dass auch ihre Vorgängeraktivitäten (z.B. c2) beendet sein müssen. Dies entspricht dem bereits in Abschnitt 3.1 definierten Verhalten.

- Die Kanten zwischen den Aktivitäten d3 bzw. d4 und dem Join sind optional. Deshalb kann die Akt. e (vorzeitig) gestartet werden, bevor Aktivitäten d3 und d4 beendet wurden.
  Da in Akt. d4 eine optionale Kante mündet, muss auch ihre Vorgängerakt. c4 hierfür noch nicht beendet sein.
- Zwischen Akt. d5 und dem Join befindet sich eine Soft-Kante. Deshalb muss zwar die Akt. d5 selbst, aber nicht ihre (mittels einer optionalen Kante verbundene) Vorgängerakt. c5 beendet sein, um die Akt. e vorzeitig starten zu können (vgl. Abschnitt 3.2).[6]
- Der Strictness Typ der Kante zwischen dem Join und Akt. e wird ignoriert: Das Join-Gateway selbst führt zu keiner Verzögerung, weil es sofort und automatisch „durchschaltet", sobald dies die Ausführungszustände seine Vorgängeraktivitäten d1 bis d5 erlauben. Deshalb ist irrelevant, ob Akt. e auf diesen Schaltvorgang wartet. Das gewünschte Verhalten bzgl. einer vorzeitigen Ausführbarkeit der Akt. e wird durch die in den Join eingehenden Kanten detailliert definiert (für jeden Pfad einzeln). Dieses soll nicht durch den Strictness-Typ der einzelnen aus dem Join ausgehenden Kante „überschrieben" werden. Dies gilt generell für die Kante nach einem Join-Gateway und ebenso für in einen Split-Gateway mündende Kanten (in Abb. 4 die Kante von Akt. a zum OR-Split).[7]

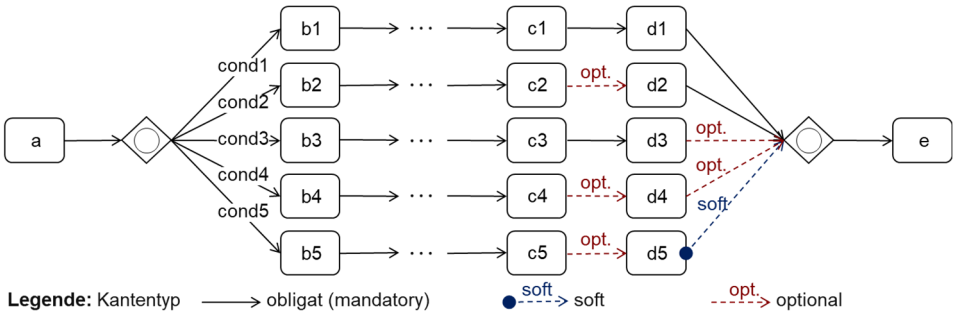

Abb. 4: OR-(Inklusives) Gateways mit Kanten verschiedenen Strictness-Typen

Der Gateway-Typ (XOR, OR, AND) legt ausschließlich fest, wie viele der Pfade ausgeführt werden. Er hat keinen Einfluss darauf, ob auf die Bearbeitung von Aktivitäten

[6] Abb. 4 enthält kein analoges Beispiel mit einer in Akt. d5 eingehenden Mandatory-Kante, weil (wie in Abschnitt 3.2 erwähnt) auf diesem Strictness-Typ folgende Soft-Kanten dasselbe Verhalten haben, wie Mandatory-Kanten. Das Verhalten wäre also identisch zum oberen Pfad mit den Aktivitäten c1 und d1.

[7] Da der Strictness-Typ dieser Kanten irrelevant ist, werden in den Abbildungen stets klassische Mandatory-Kanten verwendet. Dies hat keine Auswirkungen auf die in Abschnitt 4.2 entwickelte Semantik, weil diese Kanten in den Ausführungsregeln nicht verwendet werden (sondern nur die in den Join mündenden Kanten). Dass diese Kanten ignoriert werden, mag irritierend wirken. Alternativ könnte, so wie für Akt. c in Abb. 3 dargestellt, auch ein GP-Metamodell gewählt werden, bei der sich zwischen einem Join und der nachfolgenden Aktivität gar keine Kante befindet, sondern der Gateway-Typ (XOR, OR, AND) eine Markierung dieser Aktivität darstellt (vgl. ADEPT [RD98]). Da sich der BPMN-Standard aber gegen solch eine Notation entschieden hat, wird diese Darstellungsart im Folgenden nicht verwendet.

gewartet werden soll, weil in klassischen GP-Metamodellen auf die Beendigung aller ausgeführten Pfade gewartet wird. Deshalb ist dieses Verhalten bei anderen Gateway-Typen identisch zum eben erläuterten OR-Join. Ein analoges Verhalten ergibt sich bei verschachtelten Strukturen, z.B. wenn in einem Pfad nach einem OR-Split ein AND-Split folgt. Auch bei solch komplexen Strukturen entstehen durch den vorgestellten Ansatz keine prinzipiell neuen Problemstellungen, weil auch heutige Prozess-Management-Systeme (PMS) bereits analysieren können, welche Pfade bei dieser Prozessinstanz tatsächlich ausgeführt werden, so dass auf diese gewartet werden muss. Diese Fähigkeit von PMS nutzen die Def. 3 und 4 in Abschnitt 4.1.

Es ist nicht möglich, Soft-Kanten (vgl. Abb. 5a) durch Gateways zu ersetzen. Im GP aus Abb. 5b suggeriert die von Akt. a zum Join-Gateway verlaufende optionale Kante, dass nicht auf die Beendigung der Akt. a gewartet werden muss. Allerdings muss bei dem OR-Join, wegen der von Akt. b aus eingehenden Mandatory-Kante, auf die Beendigung aller Vorgängeraktivitäten gewartet werden (vgl. BPMN: lediglich überhaupt nicht ausgeführte Pfade werden von einem Join-Gateway ignoriert). In dem Beispiel aus Abb. 5b ist die Akt. a jedoch eine tatsächlich ausgeführte Vorgängeraktivität von Akt. b, so dass das Gateway auf diese wartet (trotz der Optionalität der Kante von Akt. a zum Gateway). Dies ist im Original-GP (Abb. 5a) jedoch nicht erwünscht. Auch die Verwendung einer optionalen Kante zwischen der Akt. b und dem OR-Join würde keine Lösung darstellen, weil dann nicht mehr auf die Beendigung der Akt. b gewartet wird (was jedoch erforderlich sein soll, vgl. Soft-Kante in Abb. 5a). Deshalb ist der Strictness-Typ soft tatsächlich erforderlich, um alle gewünschten Arten des Verhaltens realisieren zu können.

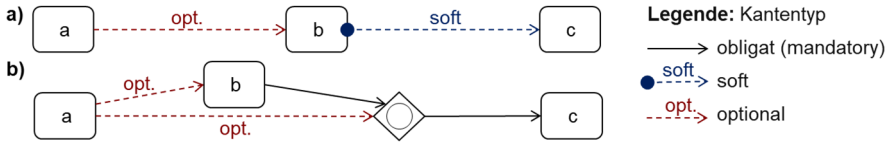

Abb. 5: Beispiel zur Erläuterung der Notwendigkeit von Soft-Kanten

4 Ausführungssemantik

Der nun folgende Unterabschnitt enthält einige grundlegenden Definitionen und es werden die bei optionalen Kanten zusätzlich erforderlichen Ausführungszustände für Aktivitäten eingeführt. Beides bildet die Basis, um im Abschnitt 4.2 eine formale Ausführungssemantik für optionale Kanten zu definieren. Hierzu werden Ausführungsregeln definiert, die auf Benutzeraktionen und Zuständen von Aktivitäten basieren.

Die alternativ mögliche Vorgehensweise, die Ausführungssemantik mittels eines bereits existierenden Formalismus (z.B. Petri-Netze) zu definieren, wurde nicht gewählt, weil der resultierende Ansatz unmittelbar zur Realisierung einer GP-Engine nutzbar sein soll. Existierende (z.B. kommerzielle) GP-Engines basieren häufig auf BPMN und den zugehörigen Ausführungsregeln, welche um die nachfolgend vorgestellten Regeln erweitert

werden können. Sie basieren jedoch nicht zwangsläufig auf Petri-Netzen o.ä., weshalb eine darauf aufbauende Ausführungssemantik nicht einfach zu integrieren wäre.

4.1 Definitionen und Ausführungszustände von Aktivitäten

Im Folgenden werden die für GP üblicherweise verwendeten (z.B. in [RD98, We19, OMG11]) Definitionen und die Menge der Ausführungszustände von Aktivitäten-instanzen so erweitert, dass sie als Grundlage für die Erstellung von Ausführungsregeln geeignet sind, welche die neu eingeführten Strictness-Typen berücksichtigen.

Def. 1: Ein Prozesstemplate PT=(N, E) besteht aus einer Menge von Knoten N (engl.: Node) und einer Menge von Kontrollflusskanten E (Edge).
Eine Prozessinstanz PI=(PT, State) besteht aus einem Prozesstemplate PT und einer Menge von Aktivitätenzuständen (State). Jede Aktivität $a \in N$ kann einen unterschiedlichen Ausführungszustand State(a) haben. Aktivitäten können die in Abb. 6 dargestellten Zustände einnehmen, d.h.:
$\forall a \in N$ gilt: State(a) $\in$ {Inactive, Active, Running, Completed, PreActive, PreRunning, PreCompleted}.

Das PMS kennt den aktuellen Zustand jeder Aktivität und verändert diese Zustände während der Ausführung einer Prozessinstanz mittels vorgegebener Ausführungsregeln (siehe Abschnitt 4.2). Dadurch kann das PMS die Ausführungsreihenfolge der Aktivitäten steuern, Einträge in Arbeitslisten einfügen, Aktivitätenprogramme (z.B. Formulare) starten, automatisch ausgeführte Services aufrufen, etc.

Def. 2: Eine Kontrollflusskante $e \in E$ ist definiert als
e = (SourceAct, TargetAct, Strictness) mit:
SourceAct: die Startaktivität der Kante, mit SourceAct $\in N$
TargetAct: die Zielaktivität der Kante, mit TargetAct $\in N$
Strictness: der Strictness-Typ der Kante, mit Strictness $\in$ {optional, soft, mandatory}

Um diese Informationen in den Ausführungsregeln nutzen zu können, kann mit gleichnamigen Funktionen auf sie zugegriffen werden. So liefert z.B. SourceAct(e) die Startaktivität der Kante e und State(a) den aktuellen Ausführungszustand der Akt. a. Die nachfolgenden beiden Definitionen dienen dazu, die in Abschnitt 4.2 vorgestellten Regeln leichter verständlich formulieren zu können. Sie spezifizieren die relevanten Vorgängeraktivitäten einer Akt. a, wobei bei Def. 4 nur bestimmte Kantentypen berücksichtigt werden. Die Akt. a kann mehrere direkte Vorgängeraktivitäten V_{all} haben, wenn sich zwischen diesen und Akt. a ein Join-Gateway (AND, OR oder XOR) befindet. Wie in Abschnitt 3.3 bereits erläutert (vgl. Abb. 4), sind bei einem Join nur solche Vorgängeraktivitäten $v \in V$ mit $V \subseteq V_{all}$ relevant, die in Pfaden liegen, die aufgrund der Bedingung $cond_i$ überhaupt ausgeführt werden. Eine Analyse, welche Pfade tatsächlich ausgeführt werden, muss ein PMS auch bei klassischen GP-Metamodellen durchführen (vgl. BPMN 2.0 [OMG11]). Dieser Aspekt ist also unabhängig von den in diesem

Beitrag betrachteten Neuerungen, so dass dies in den nun folgenden Definitionen nur informell spezifiziert wird (Bedingung ii). Um die Lesbarkeit weiter zu erhöhen, werden in Definitionen 3 und 4 die Gateway-Knoten selbst ebenso ignoriert, wie die aus einem Join-Gateway ausgehende Kante[8], weil deren Strictness-Typ ohnehin irrelevant ist (vgl. Abschnitt 3.3). So wird für die nachfolgenden Definitionen beispielsweise das Prozessmodell aus Abb. 4 so interpretiert, als wenn es direkte Kanten von den Aktivitäten d1 bis d5 zur Akt. e gäbe.

Def. 3: RelevantPredecessors(a) beschreibt die Menge der direkten Vorgängeraktivitäten der Akt. a, die aufgrund der Gateway-Struktur des Prozesses und der Kantenbedingungen bei dieser Prozessinstanz tatsächlich ausführbar sind. Dabei beschreibt die Menge V_{all} alle Vorgängeraktivitäten, noch unabhängig von ihrer tatsächlichen Ausführbarkeit:
i) $V_{all} = \{ V' \subseteq N \mid \forall v' \in V' \text{ gilt: } \exists e \in E \text{ mit } v' = \text{SourceAct}(e) \wedge a = \text{TargetAct}(e)\}$
ii) $\text{RelevantPredecessors}(a) = \{V \subseteq V_{all} \mid \forall v \in V$ gilt, dass v bei dieser Prozessinstanz tatsächlich bereits ausgeführt wurde oder noch ausgeführt werden wird$\}$

Während mit der durch Def. 3 definierten Funktion alle relevanten (d.h. tatsächlich ausgeführten) Vorgängeraktivitäten ermittelt werden, berücksichtigt Def. 4 zusätzlich die Strictness-Typen. Hierzu wird ein bestimmter Typ s vorgegeben und es werden nur über entsprechende Kanten verbundene Aktivitäten berücksichtigt. So ergeben sich für das Beispiel aus Abb. 4 mit RelevantPredecessors(e) alle ausführbaren Vorgängeraktivitäten d1 bis d5 der Akt. e, während z.B. mit $\text{RelevPredWithEdge}_{\text{optional}}(e)$ ausschließlich die Aktivitäten d3 und d4 ermittelt werden.

Def. 4: $\text{RelevPredWithEdge}_s(a)$ beschreibt die Menge V der direkten Vorgängeraktivitäten der Akt. a, wobei nur Kanten mit der vorgegebenen Strictness s berücksichtigt werden:
i) $V_{all} = \{ V' \subseteq N \mid \forall v' \in V' \text{ gilt: } \exists e \in E \text{ mit } v' = \text{SourceAct}(e) \wedge a = \text{TargetAct}(e) \wedge \text{Strictness}(e)=s\}$
ii) $\text{RelevPredWithEdge}_s(a)$ ist identisch zu RelevantPredecessors(a) definiert (vgl. ii) in Def. 3)

Bei klassischen GP-Metamodellen wie BPMN 2.0 [OMG11] befinden sich beim Start einer Prozessinstanz alle Aktivitäteninstanzen im Zustand Inactive. Eine Ausnahme bilden die am GP-Anfang liegenden Aktivitäten: diese starten mit dem Zustand Active. Bei den anderen Aktivitäten wechselt der Zustand zu Active (vgl. 1 im oberen Zweig von Abb. 6), wenn die Bearbeitung ihrer Vorgängeraktivitäten abgeschlossen wurde. Dann wird eine solche Aktivität in die Arbeitslisten der Benutzer eingetragen und wechselt in den Zustand Running, sobald einer von diesen mit der Bearbeitung beginnt. Wird diese

[8] Die zur Umformung in einen Graphen ohne Gateways erforderliche Transformation ist recht offensichtlich, so dass diese hier nicht detailliert beschrieben wird. Auf die Verwendung eines GP-Metamodells, das ganz auf Gateway-Knoten und daraus ausgehende Kanten verzichtet (vgl. Akt. c in Abb. 3 und ADEPT [RD98]), wird – wie bereits erwähnt – verzichtet, weil diese auch beim Standard BPMN existieren.

beendet, so findet ein Zustandswechsel zu Completed statt, so dass die Nachfolgeraktivitäten bearbeitbar werden, also selbst zu Active wechseln.

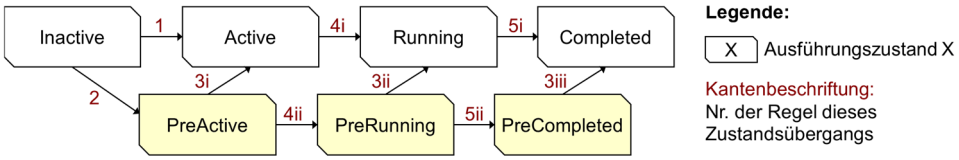

Abb. 6: Aktivitätenzustände und Zustandsübergänge

Die Menge der Zustände einer Aktivitäteninstanz wurde gegenüber solchen klassischen Ansätzen erweitert, um die neuen Strictness-Typen realisieren zu können. Die zusätzlich erforderlichen Zustände sind in Abb. 6 farbig hinterlegt und ihre Bezeichnungen beginnen „Pre“. Diese Zustände haben die Bedeutung, dass die entsprechende Akt. a „vorzeitig gestartet“ wurde, also nur ausführbar ist, weil eine eingehende Kante den Strictness-Typ optional oder soft hat. Sobald ihre Vorgängeraktivitäten regulär beendet wurden, wäre die Akt. a auch regulär startbar, so dass sie in den gleichnamigen Zustand ohne „Pre“ wechselt (3i bis 3iii in Abb. 6, die Zahlen an den Zustandsübergängen verweisen auf die Nummer der zugehörigen Regel aus Abschnitt 4.2).

In diesem Beitrag steht das Verhalten für die neuen Strictness-Typen im Fokus. Deshalb werden keine Zustände oder Regeln betrachtet, die hierfür nicht relevant sind. Solche sind z.B. für einen Abbruch einer Aktivität, deren Kompensation, erfolglos beendete Aktivitäten (Zustand Failed in BPMN 2.0 [OMG11]) oder bei Sprüngen zu anderen Aktivitäten einer Prozessinstanz [Ba22] erforderlich.

4.2 Ausführungsregeln

Der Startzustand Inactive einer Aktivität(eninstanz) wechselt zu Active (1 in Abb. 6), sobald alle relevanten Vorgängeraktivitäten (vgl. Def. 3) beendet wurden. Da es sich um den Zustandswechsel zu Active – d.h. nicht zu PreActive – handelt, müssen alle Vorgängeraktivitäten tatsächlich den Endzustand Completed erreicht haben und der Strictness-Typ der Kanten spielt keine Rolle (dieser wird in Regel 2 berücksichtigt). Konkret bedeutet das, dass jede Vorgängerakt. v der Akt. a regulär beendet sein muss, also den Zustand Completed erreicht hat. Es genügt nicht, dass sie den Zustand PreCompleted erreicht hat, d.h. bei dem GP aus Abb. 4 ist Akt. e ist erst dann regulär startbar, wenn alle ausgeführten Vorgängeraktivitäten d1 bis d5 regulär beendet wurden.

Regel 1: Wenn für eine Akt. $a \in N$ mit State(a) = Active gilt:
$\forall$ v $\in$ RelevantPredecessors(a) ist State(v) = Completed,
dann wird State(a) auf Active gesetzt.

Ist der eben beschriebene Zustandsübergang nicht möglich, so kann eine Akt. a dennoch startbar werden (und zwar vorzeitig), falls Kanten mit den Strictness-Typen optional oder soft in sie münden. Damit die „Vorzeitigkeit“ erkennbar ist, wechselt die Akt. a

dann in den Zustand PreActive (anstatt Active). Bei optionalen Kanten genügt es, dass deren Quellakt. v startbar (Zustand Active) oder selbst vorzeitig startbar (PreActive) ist, oder einen Nachfolgezustand hiervon erreicht hat, d.h. der Zustand von Akt. v darf lediglich nicht Inactive sein. Bei Soft-Kanten muss die Bearbeitung der Quellakt. v beendet sein, wobei jedoch die Vorgänger von Akt. v noch nicht beendet sein müssen. Das ist daran erkennbar, dass sich die Akt. v im Zustand PreCompleted befindet (d.h. nach einer vorzeitigen Bearbeitung beendet wurde). Natürlich ist ein solcher Zustandswechsel der Akt. a auch möglich, wenn eine bestimmte Vorgängerakt. v regulär beendet wurde (Zustand Completed). Alle über Mandatory-Kanten verbundenen Vorgängeraktivitäten müssen, so bei Regel 1, beendet sein, also den Zustand Completed erreicht haben.

Regel 2: Wenn für eine Akt. $a \in N$ mit State(a) = Inactive gilt:
Die Bedingung von Regel 1 ist nicht erfüllt $\wedge$
$\forall\ v \in \text{RelevPredWithEdge}_{\text{mandatory}}(a)$ ist State(v) = Completed $\wedge$
$\forall\ v \in \text{RelevPredWithEdge}_{\text{soft}}(a)$ ist State(v) $\in$ {Completed, PreCompleted} $\wedge$
$\forall\ v \in \text{RelevPredWithEdge}_{\text{optional}}(a)$ ist State(v) $\neq$ Inactive,
dann wird State(a) auf PreActive gesetzt.

Eine Akt. a befindet sich in einem Pre-Zustand, wenn sie startbar ist, obwohl nicht alle direkten Vorgängeraktivitäten beendet wurden. Erreichen all diese irgendwann später den Zustand Completed, dann wurden sowohl diese, als auch ihre Vorgängeraktivitäten (d.h. die indirekten Vorgängeraktivitäten der Akt. a) regulär beendet. Deshalb kann die Akt. a in den entsprechenden Zustand ohne „Pre“ wechseln. Befand sich Akt. a im Zustand PreActive (i in Regel 3), dann ist sie ab jetzt regulär startbar, d.h. die Warnung an den Benutzer entfällt (wie bei Abb. 1b erläutert entfällt die Warnung für die Akt. Röntgen, sobald die Akt. EKG beendet wurde). Die nachfolgende Regel definiert alle Zustandsübergänge vom unteren zum oberen Pfad von Abb. 6.

Regel 3: Wenn für eine Akt. $a \in N$ mit State(a) $\in$ {PreActive, PreRunning, PreCompleted} gilt:
$\forall\ v \in$ RelevantPredecessors(a) ist State(v) = Completed,
dann ergibt sich der neue Zustand dieser Akt. a folgendermaßen:

$$\text{State}(a) = \begin{cases} \text{Active}, & \text{falls State(a)=PreActive war} \quad \text{(i)} \\ \text{Running}, & \text{falls State(a)=PreRunning war} \quad \text{(ii)} \\ \text{Completed}, & \text{falls State(a)=PreCompleted war} \quad \text{(iii)} \end{cases}$$

Wenn eine Akt. a vom Benutzer gestartet wird, dann wechselt sie normalerweise in den Zustand Running (vgl. BPMN 2.0 [OMG11]). Im Kontext optionaler Kanten wird zusätzlich der Zustand PreRunning erforderlich. Durch diesen wird erkannt, dass die Akt. a noch nicht regulär startbar war, sondern vorzeitig gestartet wurde. Analog wechselt sie später (Regel 5) nicht in den Zustand Completed, sondern PreCompleted.

Regel 4: Wenn die Akt $a \in N$ einen Zustand State(a) $\in$ {Active, PreActive} hat, dann kann diese von einem Benutzer gestartet werden. Ihr Zustand ändert sich dann in:

$$\text{State(a)} = \begin{cases} \text{Running,} & \text{falls State(a)=Active war} \quad \text{(i)} \\ \text{PreRunning,} & \text{falls State(a)=PreActive war} \quad \text{(ii)} \end{cases}$$

Eine aktuell ausgeführte Akt. a kann vom Bearbeiter abgeschlossen werden. Wurde sie vorzeitig gestartet (d.h. PreRunning), dann geht ihr Zustand nicht direkt in Completed über, damit erkennbar ist, dass die Bearbeitung vorzeitig erfolgt ist. Dadurch bleibt z.B. eine Nachfolgerakt. n, die mit einer nicht-optionalen Kante verbunden ist (d.h. Strictness-Typ mandatory oder soft), nicht startbar, weil die Regel 1 zwar den Zustand Completed berücksichtigt, nicht aber PreCompleted.

Regel 5: Wenn die Akt $a \in N$ einen Zustand State(a) $\in$ {Running, PreRunning} hat, dann kann diese von einem Benutzer beendet werden. Ihr Zustand ändert sich dann in:

$$\text{State(a)} = \begin{cases} \text{Completed,} & \text{falls State(a)=Running war} \quad \text{(i)} \\ \text{PreCompleted,} & \text{falls State(a)=PreRunning war} \quad \text{(ii)} \end{cases}$$

5 Zusammenfassung und Ausblick

Wie in Abschnitt 2 dargestellt, wird die Relevanz optionaler Kanten in der wissenschaftlichen Literatur zwar erwähnt, aber ihr gewünschtes Verhalten wurde noch nicht exakt definiert. Dies betrifft sowohl das Verhalten des PMS gegenüber den Benutzern (z.B. das Anbieten von Einträgen in Arbeitslisten) als auch das der GP-Engine zur Runtime. Bei der Analyse dieser Forschungslücke wurde erkannt, dass das erforderliche Verhalten keineswegs eindeutig ist, wenn nicht nur eine einzelne optionale Kante betrachtet wird, sondern eine ganze Sequenz von derart verbundenen Aktivitäten. Deshalb wurde der zusätzliche Strictness-Typ „Soft“ eingeführt und in den Ausführungsregeln geeignet behandelt. Diese berücksichtigen zudem, dass sich vor einem Join-Gateway mit unterschiedlichen Strictness-Typen verbundene Aktivitäten befinden können.

Die Strictness-Typen verursachen kaum Aufwand bei der GP-Modellierung, weil sicherlich die meisten Kanten weiterhin mandatory sein werden, was die Standardeinstellung des Modellierungswerkzeugs sein sollte. Da zur Wahl der anderen Typen z.B. lediglich deren Typ ausgewählt werden muss, sind sie für GP-Designer sehr einfach verwendbar. Endanwender müssen lediglich einen Hinweis in ihrer Arbeitsliste beachten und können solche Aktivitäten ansonsten wie gewohnt bearbeiten. Deshalb stellen die neuen Kantentypen für beide Personengruppen keine Herausforderung dar, welche eine Evaluation der „Usability“ erforderlich machen würde. Eine technische Evaluation der neu vorgestellten Ausführungsregeln mittels eines Prototyps steht aktuell noch aus. Da diese den klassischen Ausführungsregeln einer GP-Engine stark ähneln, ist hierbei jedoch nicht mit besonderen Erkenntnissen zu rechnen.

Literaturverzeichnis

[APS09] Aalst, W. van der; Pesic, M.; Schonenberg, H.: Declarative Workflows: Balancing between Flexibility and Support. Computer Science-Research and Development 23, S. 99–113, 2009.

[AWG05] Aalst, W. van der; Weske, M.; Grünbauer, D.: Case Handling: A New Paradigm for Business Process Support. Data & Knowledge Engineering 2/53, S. 129–162, 2005.

[Ba19] Bauer, T.: Pre-modelled Flexibility for Business Processes. Proc. 21th Int. Conf. on Enterprise Information Systems, S. 547–555, 2019.

[Ba20] Bauer, T.: Business Processes with Pre-designed Flexibility for the Control-Flow. Proc. 22nd Int. Conf. on Enterprise Information Systems, 631–642, 2020.

[Ba21] Bauer, T.: Pre-modelled Flexibility for the Control-Flow of Business Processes: Requirements and Interaction with Users. Enterprise Information Systems, 833–857, Springer, 2021.

[Ba22] Bauer, T.: Requirements for Dynamic Jumps at the Execution of Business Processes. Proc. 12th Int. Symposium on Business Modeling and Software Design, S. 35–53, 2022.

[Ba23] Bauer, T.: Modelling of Advanced Dependencies Between the Start and the End of Activities in Business Processes. Proc. 25th Int. Conf. on Enterprise Information Systems, S. 457–465, 2023.

[Gr00] Grefen, P. et al.: CrossFlow: Cross-organizational Workflow Management in Dynamic Virtual Enterprises. Computer Systems Science & Engineering 5, S. 277–290, 2000.

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

[Li06] Li, M. et al.: DOPA: A Data-Driven and Ontology-Based Method for Ad Hoc Process Awareness in Web Information Systems. Proc. Int. Conf. on Web Information Systems Engineering, S. 114–125, 2006.

[OMG11] Object Management Group: Business Process Model and Notation (BPMN) 2.0, 2011.

[Pe07] Pesic, M. et al.: Constraint-based Workflow Models: Change Made Easy. Proc. 15th Int. Conf. on Cooperative Information Systems Vilamoura, S. 77–94, 2007.

[RD98] Reichert, M.; Dadam, P.: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems 2/10, S. 93–129, 1998.

[RH06] Russell, N.; Hofstede, A. ter: Workflow Control-Flow Patterns. A Revised View, BPM Center Report BPM-06-22, 2006.

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

[We19] Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, 2019.