# Modellierung von Entscheidungen und Interpretation von Entscheidungsoperatoren in einem WfMS

Jens Brüning, Peter Forbrig

Fakultät für Informatik und Elektrotechnik
Universität Rostock
Albert-Einstein-Str. 21
18059 Rostock
{jens.bruening, peter.forbrig}@uni-rostock.de

**Abstract:** In diesem Artikel wird die Modellierung von Entscheidungen in Workflows aus diversen Blickwinkeln betrachtet. Es existieren verschiedene Arten von Entscheidungen, die bei der Modellierung oder während des Ablaufs der Workflows getroffen werden müssen. Für die Modellierung gibt es unterschiedliche Sprachmittel bzw. Modellierungsweisen in Prozesssprachen wie z.B. EPKs, UML Aktivitätsdiagrammen oder YAWL um Entscheidungen auszudrücken. Der Prozessmodellierer kann zum einen die Art der Entscheidung und zum anderen die Phase festlegen, in der die Entscheidung getroffen werden soll. Für EPKs wird das Konzept der Entscheidungsfunktion um Entscheidungsprozesse erweitert, die es ermöglichen komplexere Entscheidungen mit Hilfe von hierarchischer Verfeinerung von Prozessen und Ereignissen zu modellieren. Des Weiteren werden implementierte Konzepte zur Endnutzer-Darstellung von unterschiedlichen Entscheidungsoperatoren an einem Workflow Management System (WfMS) vorgestellt.

## 1 Einleitung

Prozessmodelle stellen Handlungsabläufe dar, die Geschäftsprozesse dokumentieren und Abläufe vorschreiben. Es existieren unterschiedliche Modellierungsweisen von Prozessen, die Prozessmodelle entweder flexibler oder restriktiver für die Ausführung machen. Bei der Modellierungsweise *flexible by design* [SRM07] werden zwar die Aktivitäten definiert, die im Prozess ausgeführt werden müssen, die Reihenfolge ihrer Abarbeitung wird während der Designtime aber offen gelassen. Der ausführenden Person werden damit alternative Ausführungsmöglichkeiten gegeben. Diese muss sich bei jedem Schritt neu entscheiden und selektieren, welche Aktivität als nächstes durchgeführt werden soll. Da weniger vorgegeben wird hat der Nutzer zwar mehr Freiheiten, allerdings auch weniger Unterstützung. Meist kosten die Entscheidungen während der Runtime, die den Ablauf des Prozesses betreffen, auch Zeit.

Bei der Modellierung der Prozesse muss also ein Kompromiss zwischen Flexibilität und Unterstützung des Nutzers durch Vorgabe der Prozessschritte gefunden werden.

Es gibt außerdem unterschiedliche Arten von Entscheidungen, die zu unterschiedlichen Zeiten des Prozess-Lebenszyklus bzw. deren Instanzen getroffen werden können. Zum einen gibt es Entscheidungen, die die Ablauflogik der Aktivitäten betreffen. Hier werden viele Entscheidungen schon während der Designtime für das Prozessmodell getroffen, um festzulegen welche Aktivitäten nacheinander auszuführen sind. Zum anderen können während der Designtime Referenzmodelle herangezogen werden, die Entscheidungsoperatoren über Prozessalternativen beinhalten. Diese sind aufzulösen oder in die Runtime zu verlagern, um konfigurierte, unternehmensspezifische Prozessmodelle zu erhalten [Sch98].

Entscheidungen über die Wahl eines Prozesspfades können sowohl explizit als spezielle Aktivität in Geschäftsprozessen als auch implizit ohne Angabe einer solchen modelliert werden. Während der Designtime kann dabei bereits die Art der zu treffenden Entscheidung gewählt werden. Entscheidungen können von Menschen oder Maschinen getroffen werden, wobei dieser Artikel den Fokus auf die Modellierung der menschlichen Entscheidungen in Geschäftsprozessen legt.

Der Artikel ist wiefolgt gegliedert. In Abschnitt 2 werden die Grundlagen gelegt, indem die verwendeten Workflow Modellierungssprachen kurz vorgestellt werden. In Abschnitt 3 werden die unterschiedlichen Entscheidungsarten in den verschiedenen Workflowsprachen modelliert. Durch die Verwendung mehrerer Sprachen zusätzlich zu den EPKs werden die Konzepte der unterschiedlichen Entscheidungsmodellierung aus verschiedenen Blickwinkeln beleuchtet und die Sprachen zueinander in Kontrast gesetzt und Ausdrucksmöglichkeiten verglichen. Abschnitt 4 präsentiert eine Implementierung von den Entscheidungsoperatoren an einem WfMS. Schließlich wird im Abschnitt 5 die Arbeit zusammengefasst und ein Ausblick für weitere Arbeiten gegeben.

## 2 Grundlagen

Im weiteren Verlauf des Artikels werden unterschiedliche Prozessmodellierungssprachen verwendet. Es werden die EPK [KNS92], YAWL [HA05], UML2 Aktivitätsdiagramme (AD) [UML] und Aufgabenmodelle [Pa00, LPV01] verwendet. In diesem Abschnitt werden kurz die Grundlagen dafür gegeben.

Ereignisgesteuerte Prozessketten (EPKs) wurden am Institut für Wirtschaftsinformatik im Saarland entwickelt, um Geschäftsprozesse zu modellieren. Die Sprachelemente sind Funktionen und Ereignisse die über Kanten alternierend verbunden sind. Des Weiteren gibt es Konnektoren, die den Steuerfluss aufspalten und wieder zusammenführen können. Beim Aufspalten des Prozesspfads mit dem AND Konnektor werden Prozesspfade beschrieben, die voneinander unabhängig sind und auch parallel ausgeführt werden dürfen. Beim XOR-Split wird an der Funktion unmittelbar davor während der Runtime geprüft, welcher Pfad hinter dem Konnektor zu nehmen ist. Beim OR Split Konnektor können auf die gleiche Weise mehrere Pfade genommen werden.

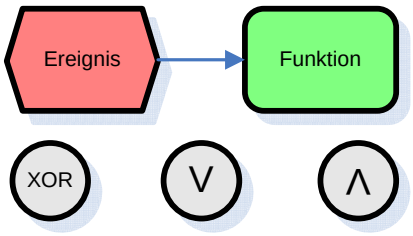



Abbildung 1: Grundelemente von EPKs

Yet another Workflow Language (YAWL) [HA05] ist eine Modellierungssprache für Workflows, die zusätzlich zum Modellierungswerkzeug (Editor) durch ein Workflow Management System (WfMS) implementiert wird. Die aus dem Editor entstandenen Modelle werden vom WfMS entgegengenommen, instanziiert und nutzergesteuert ausgeführt. Das WfMS berücksichtigt auch Daten- und Ressourcen-Aspekte, die modelliert und von der Workflow-Engine genutzt werden. Abbildung 2 zeigt die Sprachelemente. Elementare Bestandteile sind die Split- und Join-Operatoren, die direkt an die Aktivitäten notiert werden.


Abbildung 2: Grundelemente von YAWL

Die Unified Modeling Language [UML] stammt aus dem Bereich der Softwaretechnik. Mit UML lassen sich diverse Aspekte von Softwaresystemen mit unterschiedlichen Diagrammarten modellieren. Unter anderem gehören auch Aktivitätsdiagramme (AD) dazu, die auch zur Workflow-Modellierung geeignet sind. Die Grundelemente werden in Abbildung 3 veranschaulicht. Ebenfalls wie die bisher vorgestellten Sprachen sind auch ADs eine graphbasierte Sprache.

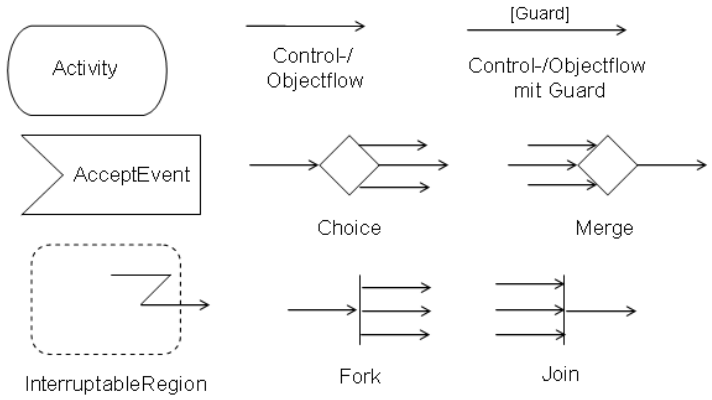



Abbildung 3: Grundelemente von UML Aktivitäts-Diagrammen

Aufgabenmodelle stammen aus dem Bereich der Human Computer Interaction (HCI) und beschreiben dort Handlungsabläufe, die ausdrücken wie der Mensch mit dem Computer interagiert. Sehr verbreitet sind hier die Concurrent Task Trees (CTT) [Pa00]. In [LPV01] werden Aufgabenmodell-Ansätze gegenübergestellt. Am Lehrstuhl Softwaretechnik der Universität Rostock wurde ein Workflow-Management System – das Task Tree Management System (TTMS) entwickelt, das auf hierarchischen Aufgabenmodellen ähnlich zu den CTTs basiert. Der Editor zum Erstellen der Workflow-Modelle stellt diese aber auch visuell ähnlich eines UML-Aktivitätdiagramms dar. Zusammenhänge zwischen CTT und Aktivitätsdiagrammen sind in [BDF07] dargestellt. In Abbildung 4 ist der Editor zum Erstellen der Workflowmodelle abgebildet. Ein Workflow als Aufgabenbaum (ohne temporale Operatoren) ist links und die Aktivitätsdiagramm-ähnliche Darstellung ist dann rechts zu sehen.

In Abschnitt 3 wird nicht auf die Modellierung von Workflows in Form von Aufgabenbäumen eingegangen. Dort wird zunächst die Modellierung von Entscheidungen in den Sprachen EPK, UML ADs und YAWL betrachtet. In Abschnitt 4, wird wieder auf diese Modellierungsart bei der Betrachtung des TTMS eingegangen. Es wird gezeigt, wie die Interpretation der unterschiedlichen in Abschnitt 3 präsentierten Entscheidungsoperatoren dort umgesetzt ist.

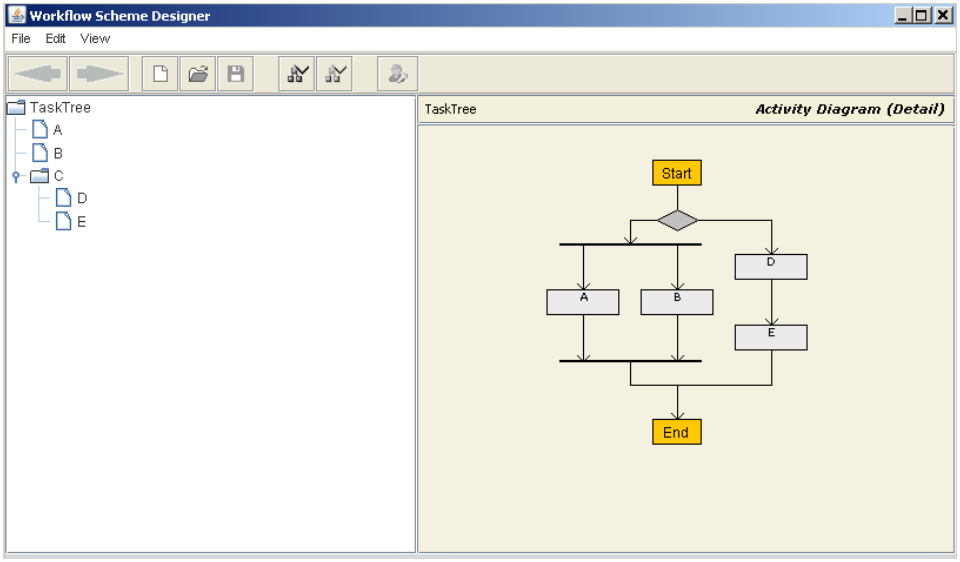



Abbildung 4: Aufgabenbaum links mit entsprechender AD ähnlicher Darstellung rechts

# 3 Modellierung von Entscheidungen

Es gibt verschiedene Arten von Entscheidungen, die während der Ausführung eines Workflows zur Runtime getroffen werden müssen. Wir unterscheiden die Arten in den folgenden Unterabschnitten: 1. können bzw. müssen Entscheidungen über den Ablauf der Aktivitäten im Geschäftsprozess getroffen werden, wenn die Reihenfolge während der Designtime noch nicht festgelegt wurde. 2. gibt es unterschiedliche Entscheidungsarten für alternativ auszuführende Aktivitäten im Prozess. Dafür gibt es unterschiedliche Entscheidungsoperatoren, die nachfolgend vorgestellt werden. 3. wird für EPKs das Konzept des Entscheidungsprozesses eingeführt, um auch komplexe Entscheidungen modellieren zu können. Schließlich sind 4. die unterschiedlichen Zeiten im Prozesslebenszyklus aufgelistet, in denen die entsprechenden Entscheidungen zu treffen sind.

## 3.1 Entscheidungen über den Prozessablauf

Es gibt Unterschiede bei Workflowmodellen bezogen auf die Ausführungsflexibilität der enthaltenen Aktivitäten. Es gibt Prozesse, die nach dem Prinzip *flexible by design* [SRM07] modelliert sind und dem Nutzer mehrere Ausführungsmöglichkeiten während der Runtime überlassen.

Nehmen wir an, dass eine Aktivitätsinstanz den Lebenszyklus aus dem UML-StateChart von Abbildung 5 hat, so dass die Aktivitäten initial in den Zustand *waiting* versetzt werden, dann gestartet und schließlich beendet werden können. Alternativ können Aktivitäten auch mit *skip* übersprungen werden.

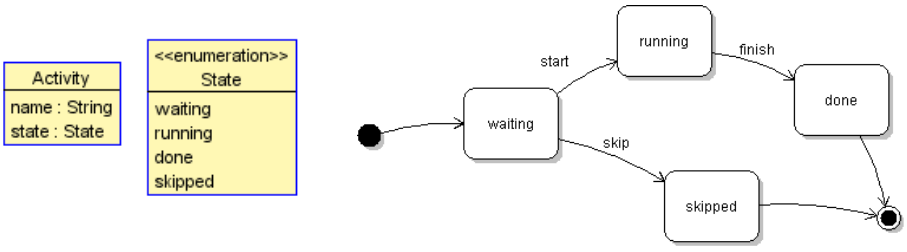



Abbildung 5: Lebenszyklus von Aktivitäten bzw. Aktivitätsinstanzen

Das EPK-Modell aus Abbildung 6 a) ist ein Beispiel für ein flexibles Ausführungsmodell. Hier können u.a. folgenden Ablaufreihenfolgen stattfinden *(start(A), start(B), finish(A), finish(B))*, *(start(A), finish(A), start(B), finish(B))*. Dagegen gibt es bei der restriktiven Prozessmodellierung aus Abbildung 6 b) nur eine Ausführungsreihenfolge *(start(A), finish(A), start(B), finish(B))*. Das zweite Modell gibt dem Nutzer mehr Unterstützung, welche Aktivität als nächstes auszuführen ist, während das erste ihm mehr Flexibilität und damit mehr Entscheidungen abverlangt.

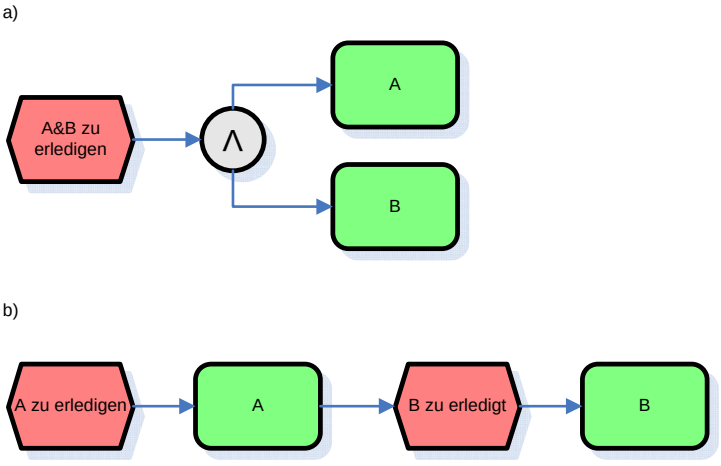



Abbildung 6: a) Flexibles Modell zur Erledigung von Aktivitäten A und B
b) restriktives Modell zur Erledingung von A und B

Es kann aber auch Situationen geben, bei denen man flexible Prozesse in der Form vom Modell aus Abbildung 6 a) haben möchte, die Flexibilität und damit den Entscheidungsraum des Nutzers für die Runtime aber etwas einschränken möchte. Der SEQ-Konnektor [ST05] ist ein Beispiel dafür. Er modelliert eine unabhängige Ausführung der nachfolgenden Aktivitäten, eine parallele Abarbeitung wird dabei jedoch untersagt.

In [ST05] wurde der SEQ-Konnektor im Zusammenhang mit einer explizit angegebenen Entscheidungsfunktion definiert, die unmittelbar vor Beginn der ersten Aktivität die komplette Ablaufreihenfolge der darauf folgenden Aktivitäten festlegt. Abbildung 7 a) zeigt ein Beispiel mit dem SEQ-Konnektor und Abbildung 7 b) den semantisch gleichen Prozessausschnitt mit XOR-Konnektoren.

Um den SEQ-Konnektor etwas flexibler zu definieren schlagen wir in Abbildung 7 c) deklarative Annotationen vor, die den Entscheidungsraum während der Runtime für flexible Prozessmodelle nicht so restriktiv einschränkt, wie die Modelle aus a) und b). In Abbildung 7 c) wird dabei mit Hilfe OCL-ähnlicher Formeln [OCL] gefordert, dass die Menge der laufenden Aktivitäten von A und B kleiner oder gleich 1 ist. Die OCL-Formel bezieht sich hierbei auf die Metaklasse *Activity* und die Enumeration *State* aus Abbildung 5. Mit dieser Notation wird die explizite Entscheidung „Über Abfolge entscheiden“ weggelassen und eine implizite Entscheidung des Nutzers zur Abfolge der Aktivitäten während der Runtime gefordert. Die Semantik ist somit gleich zum *Interleaved Parallel Routing* Pattern [DH01, S.9f]. Das Modell aus c) ist flexibler, weil hier vor der Abarbeitung der ersten Aktivität die Abfolge noch nicht feststeht, wohingegen bei a) und b) die Reihenfolge schon in der Funktion „Über Abfolge entscheiden“ festgelegt wird.

Es sind bei den deklarativen Annotationen auch andere temporale Beziehungen ausdrückbar. Beispielsweise kann man mit folgender OCL ähnlichen Formel fordern, dass wenn Funktion A läuft auch B laufen muss: *A.state=running implies B.state=running*.

Oder man fordert strikte oder auch logische Parallelität: *A.state=B.state*. Bei Workflows wird wohl häufiger die logische Parallelität vorkommen, so dass der Start und das Ende nicht exakt gleichzeitig geschehen müssen: Z.B. muss die Operation, die vom Chirurgen durchgeführt wird von der Schwester assistiert werden und umgekehrt. Strikte (realtime) Parallelität wird dabei wohl häufiger bei technischen Prozessen vorkommen. Weitergehende deklarative Ansätze zur Beschreibung von Workflows mit UML-Klassendiagrammen und OCL-Formeln sind in [BW09, Br09] zu finden.

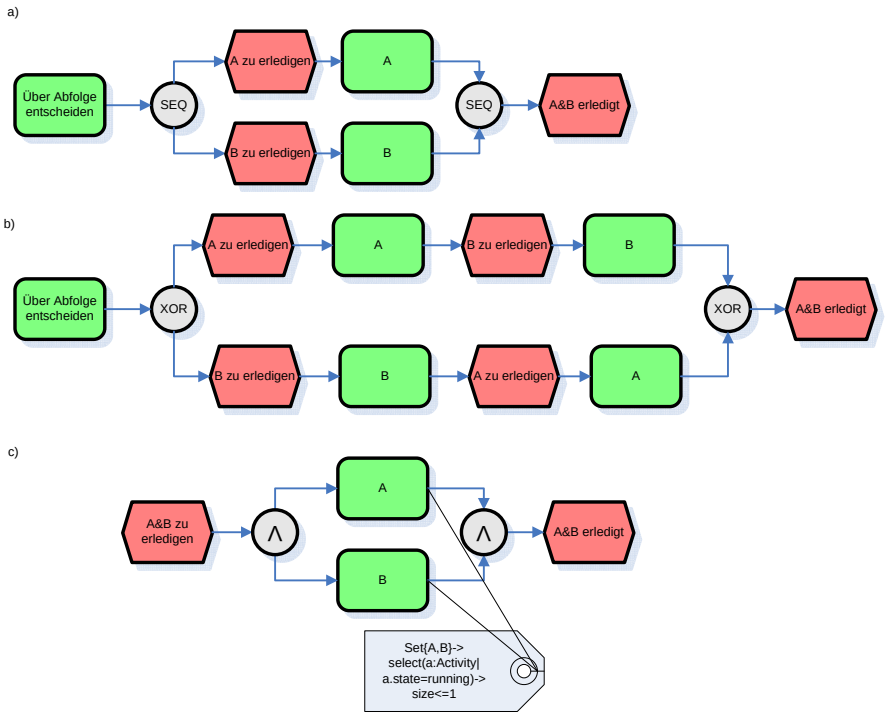



Abbildung 7: a) SEQ-Konnektor nach [ST05]
b) SEQ-Semantik ausgedrückt mit XOR-Konnektoren
c) SEQ-Semantik ausgedrückt mit deklarativen OCL-ähnlichen Annotation

### 3.2 Modellierung der ereignisbasierten und impliziten Entscheidungen

Bei den Prozess-Patterns [RH06] wird der *Deferred Choice* (Pattern 16) aufgeführt. Dieser unterscheidet sich von dem *Exclusive Choice* (Pattern 4) darin, dass die alternativ auszuführenden Aktivitäten der Umgebung angeboten werden. Die beiden Alternativen stehen in temporaler Konkurrenz zueinander und nachdem eine aktiviert wurde wird die andere aufgehoben bzw. deaktiviert. In UML-Aktivitätsdiagrammen kann dieser Sachverhalt wie in Abbildung 8 a) modelliert werden [WA04].

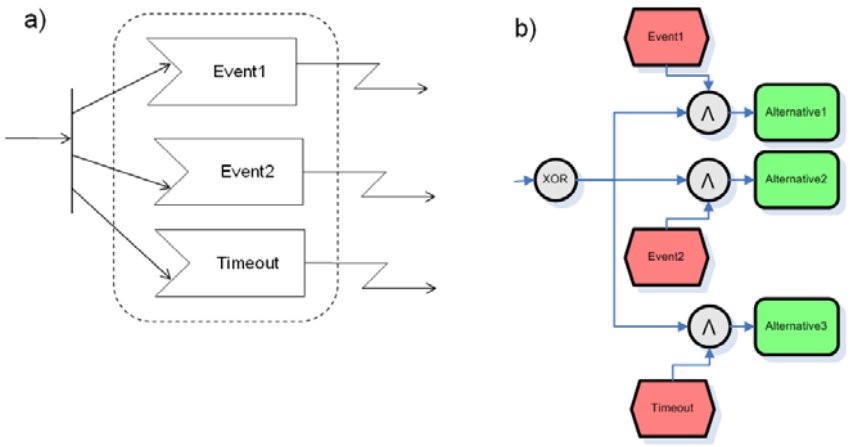



Abbildung 8: a) Modellierung des *Deferred Choice* mit UML ADs, b) Modellierung des *Deferred Choice* mit EPKs

Nach [RH06, S.133] ist der *Deferred Choice* in EPKs nicht modellierbar, weil Prozessabläufe an Konnektoren nicht verharren, sondern direkt weitergeführt werden. Außerdem bedarf es einer expliziten Entscheidungsfunktion vor XOR- bzw. OR-Split-Konnektoren [KNS92]. Lässt man diese zwei Aspekte außer acht, ist das *Deferred Choice* Pattern doch mit EPKs modellierbar. Es wird also keine explizite Entscheidungsfunktion mehr verlangt und Prozessabläufe können an Konnektoren verharren. Ein Beispiel ist in Abbildung 8 b) angegeben und eine ähnliche Modellierung wurde schon in [De02, Abb.2c)] verwendet. Die Ereignisse bei den EPKs werden dabei äquivalent zu den *AcceptEventActions* der UML Aktivitätsdiagramme verwendet. Der Ablauf verharrt bei der Prozessabarbeitung also solange am XOR-Konnektor, bis das entsprechende Ereignis eingetroffen ist. Dann ist die Bedingung für den entsprechenden AND-Konnektor erfüllt und der Kontrollfluss kann auf dem zughörigen Pfad weitergeleitet werden.

Nachdem die ereignisbasierte Entscheidung betrachtet wurde, kommen wir nun zur Modellierung der nutzerbasierten Entscheidung. Der *Deferred Choice* kann auch als eine nutzerbasierte Entscheidung interpretiert werden, bei der die Kriterien auf denen die Entscheidungen basieren im Modell nicht angegeben werden. Damit bleibt die Nutzerentscheidung unterspezifiziert und unterscheidet sich dadurch von der expliziten Entscheidung von Abschnitt 3.3. Das Verhalten des *Deferred Choice* mit der Nutzerauswahl ohne Angabe der Auswahlkriterien ist so auf der Website der Workflow-Patterns animiert [WfP].

Bei UML ADs kann für die Modellierung dieses Sachverhalts ein Choice-Operator ohne Angabe von Entscheidungskriterien in Form einer Entscheidungsfunktion und Guards für die Alternativen verwendet werden. Die Modellierung ist in Abbildung 9 a) angegeben.

In EPKs ist der *Deferred Choice* mit Nutzerauswahl ohne Angabe einer Entscheidungsfunktion ähnlich modellierbar, wie der Event-basierte *Deferred Choice* aus Abbildung 8 b). Es werden hierbei die Ereignisse weggelassen und die Kriterien zur Auswahl der Alternativen bleiben unterspezifiziert. Abbildung 9 b) zeigt diese Modellierung.

In YAWL wird für die Modellierung der Nutzerauswahl ohne Kriterien der *Condition Node* benutzt. Die alternativen Ausführungspfade werden dann mit den Aktivitäten, die mit den ausgehenden Kanten verbunden sind angegeben. Abbildung 9 c) gibt ein Beispiel dafür. Ein Event-Konzept ist in YAWL noch nicht realisiert, so dass der oben erwähnte Event-basierte *Deferred Choice* noch nicht ausdrückbar ist.

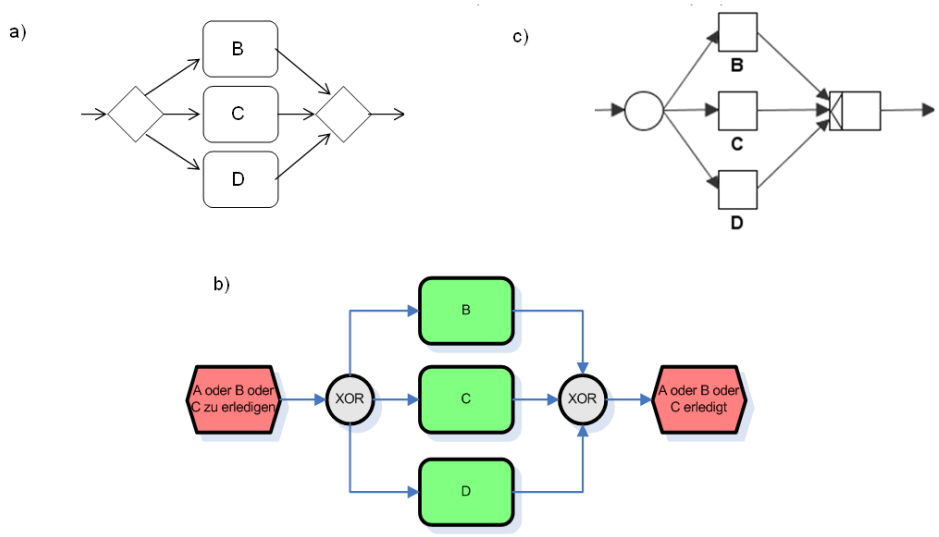



Abbildung 9: *Deferred choice* mit Nutzer Entscheidung a) UML AD, b) EPK, c) YAWL

### 3.3 Modellierung von expliziten Entscheidungen

Für die explizite Entscheidungsmodellierung benötigt man zwei Modellierungselemente, um zur Designtime die zur Runtime zu treffende Entscheidung zu modellieren. Zum einen muss ausgedrückt werden, was aktiv getan werden soll, um die Entscheidung zu treffen, z.B. Prüfen von Anträgen. Zum anderen müssen die Kriterien angegeben werden, die erfüllt sein müssen, um die entsprechenden alternativen Prozesspfade zu aktivieren. Aufgrund der angebotenen Alternativen kann der Nutzer dann die Entscheidung treffen und die dazugehörigen Pfade auswählen.

Bei UML ADs können explizite Entscheidungen mit Hilfe von Aktivitäten, dem Decision Node und Guards modelliert werden [UML, Abb.12.37] bzw. [UML, S. 359ff]. Die Aktivität, die zur Entscheidung beitragen soll (Entscheidungsfunktion) wird dabei direkt vor dem Decision Node modelliert. Hierbei ist zu beachten, dass die Bezeichnung der Aktivität zu einer Entscheidungsfunktion mit z.B. „prüfe“ oder „entscheide“ passt. An den ausgehenden Kanten hinter dem Decision Node werden Guards spezifiziert, die die Kriterien zur Aktivierung der Prozesspfade angeben. Ein Modell mit ADs ist in Abbildung 10 a) zu finden. Zusätzlich kann mit <<*DecisionInput*>> an den Decision Node eine Bedingung notiert werden, die ein wiederholtes Auswerten von Formeln in den Guards verhindert [UML, S.360].

Die explizite Entscheidung ist genau die Art der Entscheidungsmodellierung, die bei den EPKs nach [KNS92] bisher nur erlaubt war. Hierfür wurden Konnektoren XOR und OR in Verbindung mit einer Funktion, die unmittelbar vor dem Konnektor sein muss (Entscheidungsfunktion), verwendet. Des Weiteren sind die Ereignisse unmittelbar hinter den Konnektoren in das Entscheidungskonzept mit einzubeziehen. Sie stellen die Kriterien für die Auswahl dieser Pfadalternativen bereit, ähnlich zum Guard-Konzept bei den UML ADs. Abbildung 10 b) zeigt dafür ein Beispiel.

Bei YAWL gibt es die XOR- bzw. OR-Split-Operatoren, die in der Darstellung direkt an Aktivitäten geknüpft sind. Inhaltlich muss aber kein direkter Zusammenhang zwischen Aktivität und Entscheidung bestehen. Bei den Operatoren kann man über XPath-Formeln bzw. Prädikate angeben, welche Ausgangskante bzw. –kanten aktiviert werden sollen. Die Auswertung wird während der Runtime automatisch von der Workflow-Engine vorgenommen und basiert auf Daten, die vorher eingegeben wurden und über XPath abgefragt werden. In der Regel passiert die Angabe der relevanten Daten bei der Aktivität unmittelbar vor dem Operator, womit die visuelle Verknüpfung des Operators mit einer Aktivität in den meisten Fällen sinnvoll ist. Abbildung 10 c) stellt ein Beispiel dafür bereit, bei dem zu beachten ist, dass bei der Prüf-Aktivität die Ergebnisse abgefragt werden, auf der dann der darauffolgende Choice-Operator die Entscheidung automatisch vornimmt.

Bei YAWL Modellen sind auch UML-artige Guard Notationen zu finden (z.B. [LCH09]), welche aber lediglich als Kommentar dienen. Sie finden keine weitere Beachtung von der Workflow-Engine während der Runtime. Hier laufen alle expliziten Choices über das Datenmodell.

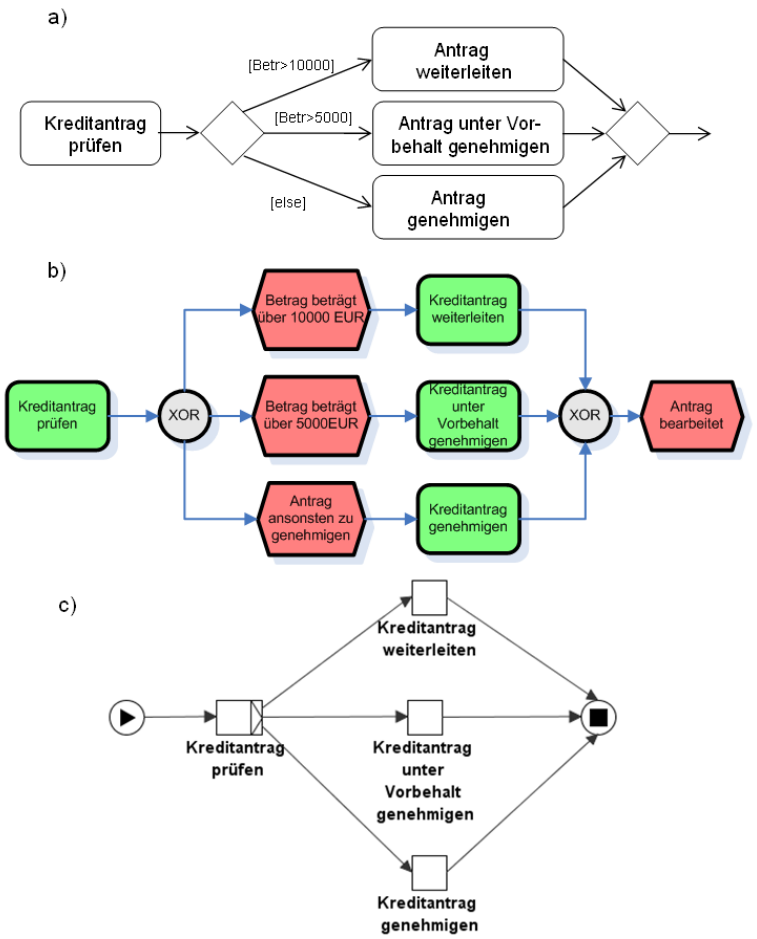



Abbildung 10: Explizite Entscheidung in folgenden Sprachen modelliert a) UML AD; b) EPK; c) YAWL

### 3.4 Modellierung von Entscheidungsprozessen

In [ST05] wurde in Abbildung 1 ein Entscheidungsprozess modelliert, der vier Entscheidungsaktivitäten umfasst. Die Abfolge der Aktivitätsabarbeitung ist in dem Modell nicht vorgegeben. Somit ist dieser Entscheidungsprozess nach dem Prinzip *flexible by design* (s. Abschnitt 3.1) spezifiziert.

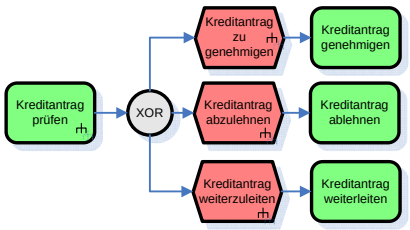
In diesem Unterabschnitt wird ein Konzept vorgestellt, um Entscheidungsprozesse zu modellieren, in denen die Ablaufreihenfolge der Entscheidungsfunktionen vorgegeben ist und die relevanten Prüfungen nicht flexibel in der Reihenfolge ausgeführt werden dürfen. Hierfür wird das Konzept der Entscheidungsfunktionen bei den EPKs, das in Unterabschnitt 3.3 vorgestellt wurde, um Entscheidungsprozesse erweitert.

Beim Konzept der Entscheidungsprozesse wird die Entscheidungsfunktion in einen Entscheidungsprozess mit Entscheidungs-Unterfunktionen untergliedert. Die Ereignisse dienen dem Nutzer als Wahlalternative während die Entscheidungsfunktion aktiv ist.

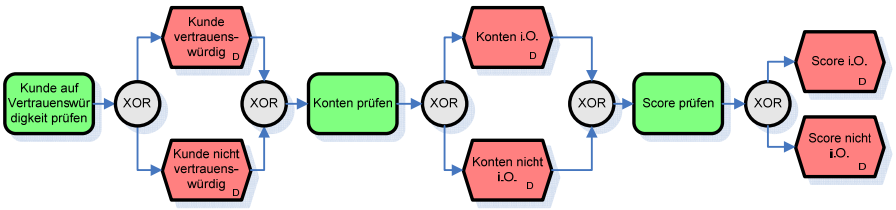
In Abbildung 11 a) wird ein Ausschnitt aus einem Prozess vorgestellt, in dem ein Kreditantrag geprüft und darüber zu entscheiden ist, ob dieser genehmigt, abgelehnt oder weitergeleitet werden muss. Sowohl die Funktion *Kreditantrag prüfen* als auch die darauffolgenden Ereignisse sind hierarchisch gegliedert, welches mit dem Hierarchie-symbol von UML ADs ausgedrückt wird.

Der Entscheidungsprozess ist in Abbildung 11 b) modelliert. Hier ist zu erkennen, dass die Prüf-Aktivitäten in der Abfolge nicht flexibel spezifiziert sind. In der Sequenz müssen hier drei Entscheidungsfunktionen abgearbeitet werden. Die Ereignisse hinter den XOR-Split-Konnektoren sind mit einem *D* gekennzeichnet, welches für *Decision* steht und bedeutet, dass diese Ereignisse für die Auswertung nach Beendigung des Entscheidungsprozesses herangezogen werden.

a)




b) Entscheidungsprozess: *Kreditantrag prüfen*



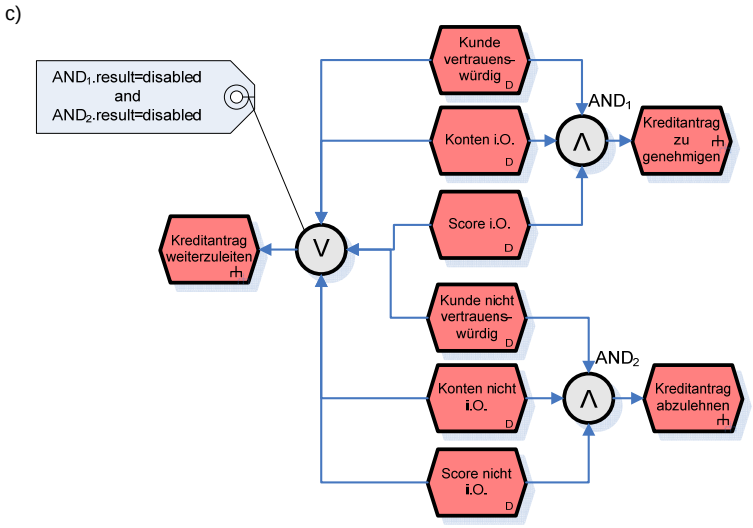



Abbildung 11: a) Übergeordneter Prozess; b) Entscheidungsprozess *Kreditantrag prüfen*; c) Regeln zum Auswerten der Entscheidungsereignisse zu den übergeordneten Ereignissen

Die Auswertungsregeln sind dann in Abbildung 11 c) ausgedrückt. Hier sind alle *Decision-Ereignisse* aufgelistet und mit logischen Ausdrücken bzw. Konnektoren verbunden, die dann schließlich zu den strukturierten Events führen. Es wird hier 1. ausgesagt, dass alle drei Prüffunktionen positiv ausfallen müssen, damit ein Kreditantrag genehmigt wird. 2. wird spezifiziert, dass bei negativer Bewertung aller Prüffunktionen der Kreditantrag abgelehnt werden muss. 3. muss bei jedem anderen Fall der Antrag weitergeleitet werden. Hierfür wurde mit einer Bedingung an dem OR-Join ähnlich zu den C-EPC-Annotationen [RA07] vermerkt, dass wenn die $AND_1$ und $AND_2$-Konnektoren nicht schalten können, also als Resultat-Zustand *disabled* liefern, der Kreditantrag weiterzuleiten ist.

### 3.5 Phasen, in denen Entscheidungen zu treffen sind

Es existieren unterschiedliche Phasen bei Geschäftsprozessmodellen, in denen Entscheidungen getroffen werden müssen. Die Entscheidungsoperatoren geben an, in welcher Phase sie aufzulösen sind oder ob sie evtl. in die nächste Phase verschoben werden können. In den folgenden Punkten werden die Phasen aufgelistet, in denen sich Geschäftsprozessmodelle befinden können.

- **Buildtime/Designtime:** Es existieren für EPKs viele Referenzmodelle [Sch98], die u.a. für unterschiedliche Branchen entwickelt wurden und z.B. mit der C-EPC [RA07] modelliert sind. In den Referenzmodellen existieren diverse sogenannte konfigurierbare Konnektoren [RA07], die während der Buildtime für das konkrete Unternehmen aufgelöst werden können. Das Referenzmodell wird dadurch für das konkrete Unternehmen konfiguriert.

  Während der Buildtime werden nur implizite Entscheidungen vom Prozess-modellierer bei Auflösung der konfigurierbaren Konnektoren gemacht. Diese haben damit den Charakter eines *Deferred Choice*, die während der Buildtime durch Designentscheidungen aufgelöst werden.

- **Instantiationtime:** In [BF08] und [APS09] wird die Instantiationtime als letzte Phase der Konfiguration der Workflow-Modelle vorgestellt, in der die Modelle für den konkreten Anwendungsfall z.B. für spezielle Kundenwünsche angepasst werden. Bei den EPKs können die konfigurierbaren Konnektoren nicht nur für Referenzmodelle angewendet werden, sondern auch durch die Instantiation-time-Operatoren bei der Instanziierung des Geschäftsprozesses [BF08].

  Bei der Instantiationtime kann eine Unterscheidung von explizit und implizit zu treffenden Entscheidungen vorgenommen werden. Einerseits können vom WfMS bei der Instanziierung die Kriterien für die zu wählenden Prozesspfade abgefragt werden, was eine explizite Entscheidung bedeuten würde. Andererseits können die möglichen Pfadalternativen ohne Kriterien direkt zur Wahl gestellt werden, was eine Art *Deferred Choice* zur Instanziierung darstellt.

- **Runtime:** Nach der Instanziierung des Geschäftsprozesses befindet man sich in der Runtime. Die in den bisherigen Abschnitten betrachteten Entscheidungs-operatoren betreffen die Runtime. Es wird zwischen expliziten und impliziten Entscheidungen unterschieden.

  Zusätzlich stellt sich während der Runtime auch noch die Entscheidung der Ablaufreihenfolge bei Prozessmodellen, die nach dem *flexible by design* Prinzip modelliert wurden. Sogar die Ablauflogik kann explizit mit einer Funktion zur Runtime entschieden werden [ST05, Abb. 4], wobei diese Modellierungsart in der Regel implizit bleibt.

## 4 Interpretation von Entscheidungsoperatoren im WfMS

In diesem Abschnitt wird das in Abschnitt 2 erwähnte TTMS (ein auf Aufgabenbäumen basierendes WfMS) näher vorgestellt. Für das WfMS wurden implizite und explizite Entscheidungsoperatoren implementiert. Im Unterschied zu YAWL laufen die Auswertungen der expliziten Entscheidungen aber nicht über das Datenmodell, sondern werden direkt über Conditions und Guards interpretiert. Es wird gezeigt, wie die Entscheidungsoperatoren interpretiert und dem Endnutzer präsentiert werden.

### 4.1 Interpretation von impliziten Entscheidungen zur Runtime

Implizite Entscheidungen werden z.B. bei *flexible by design* Modellen für die Abfolge der Aktivitäten getroffen. Hierbei sind die meisten Aktivitäten aktiviert und werden vom WfMS dem Nutzer als startbar angezeigt. Auch parallele Abarbeitung ist möglich, wenn der Nutzer neben einer schon laufenden Aktivität eine weitere startet. Interpretationen von deklarativen Annotationen, werden vom WfMS noch nicht unterstützt, so dass Modelle wie von Abbildung 7 c) noch nicht ausdrückbar sind.

Implizite Entscheidungen über Prozessalternativen sind über den *Deferred Choice* möglich und mit dem TTMS modellierbar. Beim Modell aus Abbildung 4 ist der erste Operator ein *Deferred Choice.* Der weitere Operator ist der Fork, der die Unabhängigkeit von Aktivität A und B ausdrückt. Nach Instanziierung sind drei Aktivitäten aktiviert und damit startbar: A, B und D.

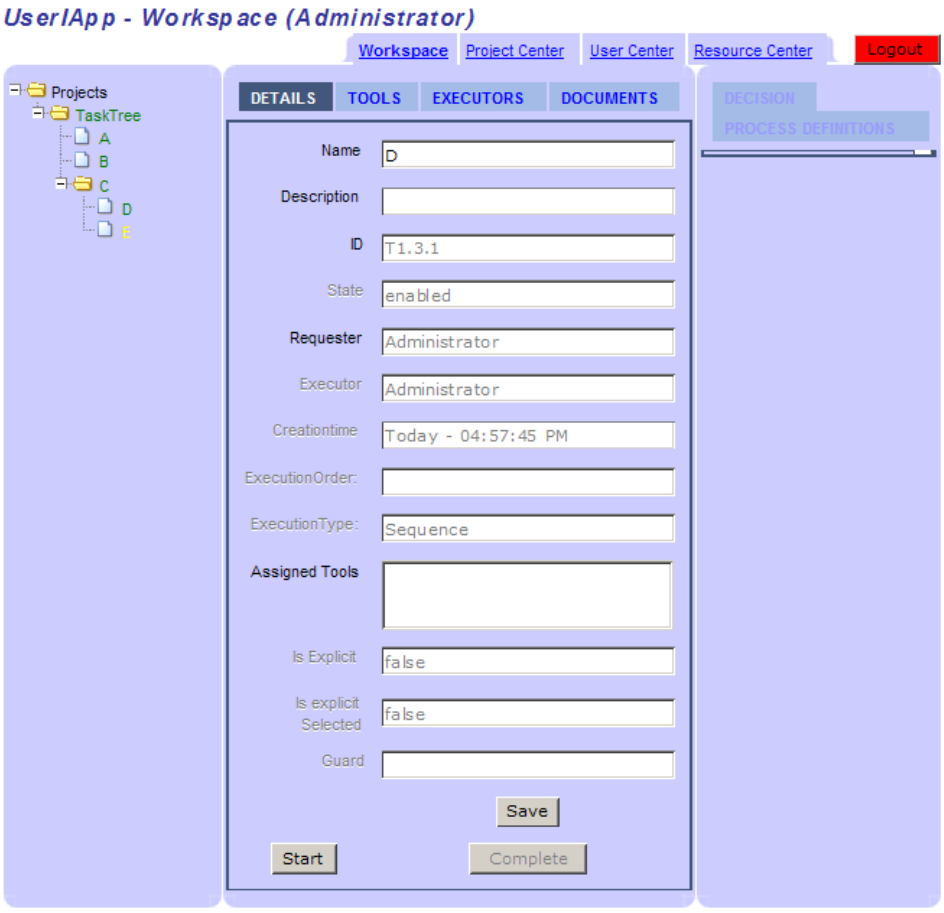



Abbildung 12: Das User Interface vom Task-Tree-Management-System (TTMS)

In Abbildung 12 ist die GUI des WfMS zu sehen, in dem die startbaren Aktivitäten grün angegeben sind. Aus Sicht des Nutzers ist hier nicht ersichtlich in welchem Verhältnis A, B und D zueinander stehen. Es könnten alle Aktivitäten unabhängig zueinander stehen oder mit einem *Deferred Choice* miteinander verknüpft sein. Startet der Nutzer nun Aktivität B, wird D automatisch deaktiviert. Wird jedoch D gestartet, werden A und B deaktiviert. In Abbildung 12 ist Aktivität D ausgewählt und startbar, was auch an dem aktiven Start-Button erkennbar ist.

### 4.2 Interpretation von expliziten Entscheidungsfunktionen zur Runtime

Um die expliziten Entscheidungen im WfMS nutzen zu können, müssen diese mit dem Editierwerkzeug modelliert werden. Abbildung 13 zeigt ein solches Modell mit einem expliziten XOR-Entscheidungsoperator inkl. Guards. Es wird eine Kreditantrags-abwicklung modelliert. Zunächst findet beim Entscheidungsoperator eine Prüfung statt. Die Bezeichnung der Entscheidungsaktivität ist dort hinterlegt. Im Anschluss werden die Kriterien zur Pfadauswahl in den Guards spezifiziert.

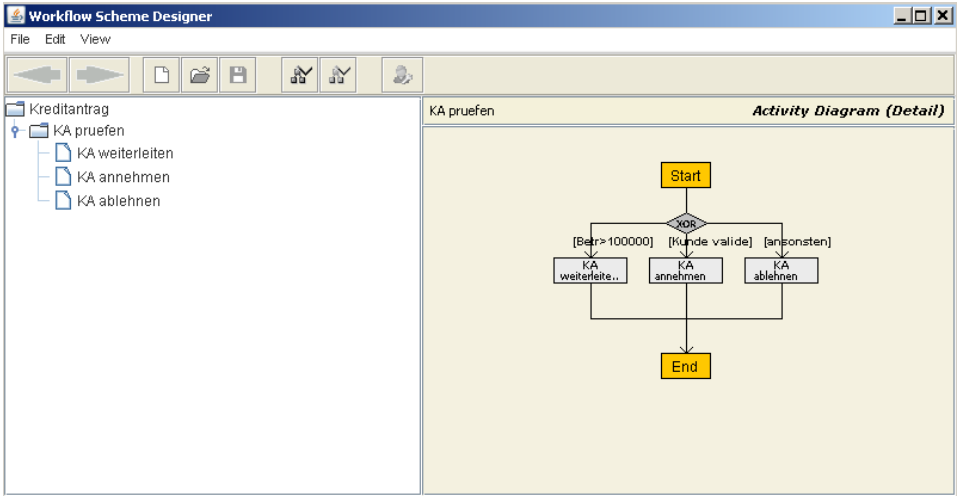



Abbildung 13: Spezifikation des expliziten Entscheidungsoperators im Editor

Abbildung 14 zeigt den Prozess während der Runtime im WfMS. Im rechten Frame ist unter den Buttons *Decision* und *Process Definitions* die Auswahl der alternativen Kriterien zu finden. Die Auswahl findet über Radio Buttons statt und garantiert dadurch eine exklusive Auswahl. Es wurde in diesem Fall das Kriterium *Kunde valide* ausgewählt, womit eine Aktivierung der dazugehörigen Aktivität erfolgt.

Beim expliziten OR-Operator werden dem Nutzer Checkboxes zur Auswahl bereitgestellt. Hier können dann mehrere Kriterien ausgewählt und Prozesspfade aktiviert werden.

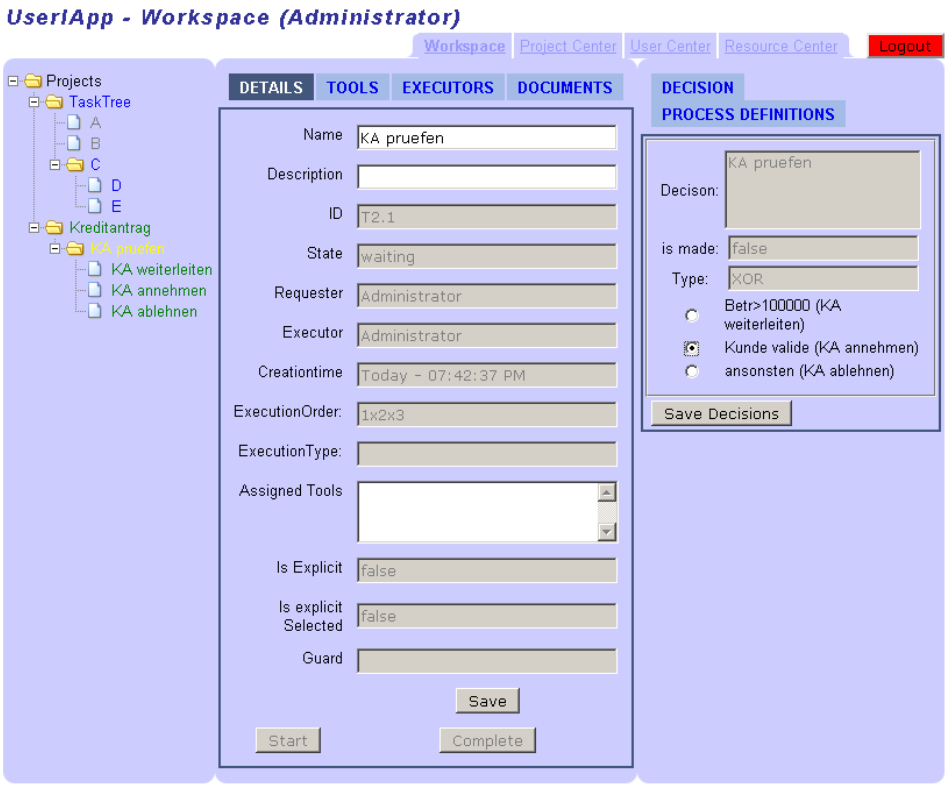



Abbildung 14: Auswertung des expliziten Entscheidungsoperators im TTMS

### 4.3 Interpretation von Instantiationtime Entscheidungen

In [BF08] wurde die Instantiationtime als Phase vorgestellt, in der die letzte Konfiguration vom WfMS vorgenommen wird. Der explizite XOR-Entscheidungsoperator aus Abbildung 13 wurde für diesen Unterabschnitt durch ein Instantiationtime XOR-Operator ersetzt. Bei der Erzeugung des Prozesses erfolgt die Auswertung dieses Operators und es erscheint der Auswertungsbildschirm von Abbildung 15. Hier kann das entsprechende Kriterium ausgewählt werden, womit der Operator aufgelöst wird. Zusätzlich kann die Entscheidung auch in die Runtime verlagert werden.


Abbildung 15: Auswertung des Instantiationtime-XOR-Operators im TTMS

## 5 Zusammenfassung und Ausblick

In diesem Artikel wurde umfassend auf Entscheidungen eingegangen, die zum einen die Ablauflogik der Workflows und zum anderen Entscheidungen zur Auswahl von Prozessalternativen betreffen. Diese Betrachtungen wurden für die unterschiedlichen Prozesssprachen EPK, UML AD und YAWL durchgeführt, die damit auch verglichen wurden. Für EPKs wurden Vorschläge unterbreitet, um auch den *Deferred Choice* ausdrücken zu können. Außerdem wurde für komplexe Entscheidungen in EPKs das Konzept der Entscheidungsprozesse eingeführt, die mehrere Entscheidungsfunktionen umfassen und in einer festgelegten Reihenfolge ablaufen sollen.

Es wurden deklarative Annotationen in OCL-Syntax vorgeschlagen, um temporale Beziehungen zwischen Aktivitäten (z.B. das *Interleaved Parallel Routing* Pattern) besser modellieren zu können. Zusätzliche temporale Beziehungen wurden vorgestellt, wie z.B. die Forderung nach der (logischen) Parallelität, die in bisherigen imperativen Prozessmodellen nicht ausdrückbar waren.

Die in [BF08] diskutierten Instantiationtime-Operatoren wurden im TTMS implementiert. Des Weiteren wurden die expliziten Entscheidungsfunktionen umgesetzt, die in Form von Conditions und Guards zur Designtime die Entscheidungskriterien festlegen können. Die implizite Entscheidung (*Deferred Choice*) war dagegen bereits Bestandteil vom TTMS.

Für weitere Arbeiten ist die Anbindung des UML-/OCL-Modellierungswerzeugs *USE* (A UML-based Specification Environment) [USE, GBR07] an das Workflow Management System TTMS wünschenswert, um deklarative Aspekte an Workflow-Abläufen überprüfen zu können. USE kann die Daten aus dem WfMS abfragen und in sein eigenes Objektmodell überführen und fortlaufend aktualisieren. Dann kann USE als Monitoring Werkzeug für Workflows benutzt werden. Beziehungen zwischen Aktivitäten können mit OCL-Invarianten deklarativ ausgedrückt werden [Br09, BW09], die dann von USE während der Runtime überprüft werden. Mit dem *Class-Invariant-View* werden dann die verletzten Invarianten hervorgehoben und mit dem *Evaluation Browser* als OCL-Expression-Debugger kann der Grund der Verletzung gefunden werden.

Außerdem stellt USE schon diverse statistische Views bereit und OCL kann in USE als Anfragesprache benutzt werden, so dass Process-Mining Techniken in Verbindung mit diesem Werkzeug angewendet werden können.

## Literaturverzeichnis


[APS09] van der Aalst, W; Pesic, M.; Schonenberg, H.; Declarative workflows: Balancing between flexibility and support. Computer Science - Research and Development, 23(2):99–113, 2009.

[BF08] Brüning, J.; Forbrig, P.: Methoden zur adaptiven Anpassung von EPKs an individuelle Anforderungen vor der Abarbeitung, EPK-Workshop2008, Saarbrücken, CEUR-WS 420

[BFD07] Brüning, J.; Dittmar, A.; Forbrig, P., Reichart, D.: Taking SW Engineers on Board: Task Modelling with Activity Diagrams, EIS2007, Salamanca, LNCS 4940, 2008

[BW09] Brüning, J.; Wolff, A.: Declarative Models for Business Processes and UI Generation using OCL, Models2009, Denver, OCL-Workshop, Electronic Communications of the EASST, http://eceasst.cs.tu-berlin.de/index.php/eceasst/issue/archive

[Br09] Brüning, J.: Declarative Workflow Modeling with UML Class Diagrams and OCL. BPSC2009, Leipzig, LNI-P 147, 2009, S. 227-228

[De02] Dehnert, J.: Making EPCs fit for Workflow Management, EPK-Workshop2002, Trier, http://www.wiso.uni-hamburg.de/fileadmin/WISO_FS_WI/EPK-Community/epk2002-proceedings.pdf, S.51-70 (besucht: 14.9.09)

[DH01] Dumas, M.; ter Hofstede; A.: UML Activity Diagrams as a Workflow Specification Language. Proceedings of the International Conference on the Unified Modeling Language (UML), Toronto, Canada, Springer, 2001.

[GBR07] Gogolla, M.; Büttner, F.; Richters, M.: USE: A UML-Based Specification Environment for Validating UML and OCL. Science of Computer Programming, 69:27-34, 2007.

[HA05] ter Hofstede, A.; van der Aalst, W.: YAWL: Yet Another Workflow Language. Information Systems 30(4), 2005, S. 245-275

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

[LPV01] Limbourg, Q., Pribeanu, C., Vanderdonckt, J.: Towards Uniformed Task Models in a Model-Based Approach. DSV-IS2001, LNCS 2220, 2001.

[LCH09] La Rosa, M.; Clemens, S.; ter Hofstede, A.: The Order Fulfilment Process Model (Appendix A1). In: ter Hofstede, H.; van der Aalst, W.; Russell, N.; Adams, M. (eds.) Modern Business Process Automation: YAWL and its Support Environment. Springer, 2009

[OCL] OMG OCL Specification v.2.0; http://www.omg.org/spec/OCL/2.0/PDF (besucht: 14.9.09)

[Pa00] Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer Verlag, 2000.

[RA07] Rosemann, M.; van der Aalst, W.: A configurable reference modelling language. Information Systems 32(1), S. 1-23, 2007.

[RH06] Russel, N.; ter Hofstede, H; van der Aalst, W.; Mulyar, N.: Workflow Patterns : A Revised View. BPM Center Report BPM-06-22, BPMcenter.org, 2006.

[Sch98] Schütte, R.: Grundsätze ordnungsgemäßer Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle. Gabler, Wiesbaden, 1998, Diss. Univ. Münster

[SRM07] Schonenberg, M.H.; Mans, R.S.; Russell, N.C.; Mulyar N.A.; v. d. Aalst W.M.P.: Towards a taxonomy of process flexibility (extended version). BPM Center Report BPM-07-11, BPMcenter.org, 2007

[ST05] Scheer, A.-W.; Thomas, O: Geschäftsprozessmodellierung mit der Ereignisgesteuerten Prozesskette. Das Wirtschaftsstudium 34, Nr. 8-9, S. 1069-1078. 2005.

[UML] OMG UML Superstructure Specification v.2.1.2 http://www.omg.org/docs/formal/07-11-02.pdf, 2007, S.295-418

[USE] A UML-based Specification Environment, Universität Bremen, Lehrstuhl Datenbanken: http://www.db.informatik.uni-bremen.de/projects/use/ (besucht: 14.9.09)

[WfP] Workflow Patterns, WCP16: Deferred Choice Pattern http://www.workflowpatterns.com/patterns/control/state/wcp16_animation.php (besucht: 14.9.09)

[WA04] Wohed, P.; van der Aalst, W.; Dumas, M.; ter Hofstede, A.; N. Russell: Pattern-based Analysis of UML Activity Diagrams**.** BETA Working Paper Series, WP 129, Eindhoven University of Technology, Eindhoven, 2004.