Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik


ulm university universität ulm

**Diplomarbeit**

# Realisierung einer Zeitmanagementkomponente eines adaptiven Prozess-Management-Systems

Andreas Lanz

28. August 2008

1. Gutachter: Prof. Dr. Manfred Reichert
2. Gutachter: Prof. Dr. Peter Dadam

Nur Wissen überlebt, nämlich
indem es den denkenden
Menschen buchstäblich informiert,
den Zustand (state) seines
Gehirns, ändert.

---

*(Joseph Weizenbaum*
*(1923-2008))*

# Kurzfassung

Prozess-Management-Systeme unterstützen eine rechnerbasierte Verwaltung und Steuerung von Geschäftsprozessen. Aufgrund des zunehmenden Wettbewerbsdrucks werden solche Systeme in den letzten Jahren für Unternehmen immer wichtiger, da sie zum einen eine Optimierung der Geschäftsprozesse erlauben und es zum anderen ermöglichen, die Mitarbeiter von lästiger Routinearbeit zu entlasten. Trotz des weltweiten Erfolgs und der vielen Vorteile, die der Einsatz eines solchen Systems mit sich bringt, gibt es immer noch einige Aspekte, die häufig gefordert, aber bisher nicht oder nur unzureichend unterstützt werden.

Ein solcher Aspekt ist die Zeitplanung/-management. Zeit spielt jedoch heutzutage eine entscheidende Rolle im Geschäftsleben. Lieferzeiten, Termine und gesetzliche Fristen müssen eingehalten, Gesamtlaufzeiten und Bearbeitungsdauern überwacht und Wiedervorlagetermine beachtet werden. Um nur einige wenige Beispiele für die Notwendigkeit von Zeitplanung und Zeitmanagement und deren Einbindung in ein Prozess-Management-System aufzuführen.

Diese Arbeit beschäftigt sich daher mit der Frage, wie ein Zeitmanagement für ein adaptives Prozess-Management-System realisiert werden kann. Ausgehend von den Anforderungen, die vonseiten der Prozess-Modellierung und der Modellierung der zeitlichen Aspekte an das Zeitmanagement gestellt werden, leiten wir die nötigen Beschreibungskonzepte ab und erweitern einen Beschreibungsformalismus für Prozesse (ADEPT) um die entsprechende Funktionalität. Auf Basis dieses erweiterten ADEPT-Metamodells werden Algorithmen vorgestellt und entwickelt, die in der Lage sind, die zeitlichen Bedingungen eines Prozesses auf deren Widerspruchsfreiheit zu überprüfen und aus diesen erlaubte Ausführungszeitpunkte für die Aktivitäten des Prozesses ableiten.

Diese Algorithmen werden im Weiteren dazu genutzt, die Zeitdaten laufender Prozesse zu aktualisieren, und darauf aufbauend, die Einhaltung der Zeitbedingungen zu überwachen. Anschließend gehen wir auf die Frage ein, wie mit Zeitbedingungen verfahren werden kann, die erst während der Ausführung der Prozesse feststehen, und welche zusätzlichen Einschränkungen für eine dynamische Änderung der Prozesse gelten. Daneben analysieren wir, welche Möglichkeiten für die Prognose der Ausführungszeitpunkte zukünftiger Aktivitäten sich aus den gewonnenen Zeitdaten ergeben und wie man diese beispielsweise für Ressourcenmanagement nutzen kann.

Um die Machbarkeit des vorgestellten Ansatzes zu demonstrieren, wurden die wichtigsten Teile der vorgeschlagenen Konzepte in ein reales Prozess-Management-System (ADEPT2) integriert.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Beispiele

# 1 Einleitung

Der wirtschaftliche und finanzielle Druck auf Unternehmen hat in den letzten Jahren, nicht zuletzt aufgrund der fortschreitenden Globalisierung, stetig zugenommen. Eine Folge hieraus ist, dass die Unternehmen gezwungen sind, ihre Geschäftsprozesse immer weiter zu optimieren. Als ein probates Mittel hierfür haben sich *Prozess-Management-Systeme* (kurz: *PMS*) erwiesen [Georgakopoulos *et al.*, 1995].

Prozess-Management-Systeme erlauben eine explizite Modellierung von Geschäftsprozessen sowie eine teilweise oder sogar vollständige Automatisierung dieser. Sie ermöglichen dabei zusätzlich eine Trennung der Ablauflogik der Prozesse vom eigentlichen Anwendungscode, wodurch sich vor allem eine Zeit- und Geldersparnis ergibt, da die Anwendungen für die einzelnen Arbeitsschritte nicht für jeden Prozess neu angepasst werden müssen. Zusätzlich unterstützt diese Trennung auch eine Gliederung des Prozesses in einfache und überschaubare Teilaufgaben.

Als Basis für ein Prozess-Management-System dient ein sogenanntes *Prozess-Metamodell*, indem die für die Modellierung der Prozesse erlaubten Konstrukte beschrieben werden. Darauf aufbauend kann ein Modellierer meist mithilfe eines graphischen Editors verschiedene Prozesse modellieren. Eine solche Modellierung eines Prozesses zur automatischen Ausführung wird als *Prozessschema* oder *Workflow* bezeichnet. Ein Prozessschema beschreibt die verschiedenen Aspekte eines Prozesses auf maschinell verständliche Weise, darunter sind unter anderem die Ablauflogik des Prozesses, die Verbindung der einzelnen Arbeitsschritte mit den dafür nötigen Anwendungen, der Datenfluss innerhalb des Prozesses und die Zuordnung von Bearbeitern und Ressourcen zu den einzelnen Arbeitsschritten. Idealerweise ist ein Prozessschema daneben auch für den Benutzer verständlich, dies hängt jedoch vom Metamodell ab. Nachdem ein Prozess durch ein solches Prozessschema vollständig spezifiziert wurde, können von diesem beliebig viele Instanzen erstellt und ausgeführt werden. Eine Instanz repräsentiert dabei eine durch das Prozess-Management-System überwachte, korrekte Ausführung des dem Prozessschema zugrunde liegenden Prozesses.

Ein wichtiger Aspekt von Prozess-Management-Systemen ist, dass die Verbindung der einzelnen Arbeitsschritte nicht mehr manuell durch die Bearbeiter durchgeführt, sondern vollständig vom Prozess-Management-System übernommen wird. Die explizite Modellierung erleichtert einerseits die Optimierung der Prozesse, beispielsweise das Auffinden doppelt durchgeführter oder überflüssiger Arbeitsschritte, andererseits werden die Prozesse hierdurch starr und der Benutzer ist nicht mehr in der Lage auf plötzlich veränderte Situationen zu reagieren. Adaptive Prozess-Management-Systeme versuchen

diesen Nachteil zu vermeiden, indem sie Möglichkeiten bereitstellen, um auf geänderte Situationen reagieren zu können. Dies bedeutet vor allem, dass einzelne Prozessinstanzen oder ganze Prozessschemata mit ihren abgeleiteten Prozessinstanzen zur Laufzeit verändert werden können [Reichert, 2000]. Dabei gehen die angebotenen Möglichkeiten weit über die reine Veränderung der Ablauflogik hinaus.

Neben der Optimierung der Geschäftsprozesse sind Prozess-Management-Systeme in der Lage, dem Endbenutzer lästige Routinearbeit abzunehmen, da diese sich beispielsweise nicht mehr um die Arbeitsverteilung und die Weitergabe der mit einem Prozess verbundenen Dokumente kümmern müssen. Hierdurch können außerdem Verzögerungen aufgrund von ungenauen Arbeitsabläufen vermieden werden. All dies lässt sich unter dem Stichwort des „intelligenten Assistenzsystems“ zusammenfassen.

Insgesamt können Geschäftsprozesse durch den Einsatz adaptiver Prozess-Management-Systemen deutlich verbessert und anpassungsfähiger gemacht werden [Dadam & Reichert, 2004]. Jedoch weisen die meisten der am Markt befindlichen Systeme noch einige Defizite auf. Beispielsweise wird die Zeit von heutigen Prozess-Management-Systemen und auch deren Prozess-Metamodellen nicht annähernd ausreichend unterstützt. Diese ist jedoch aus heutigen Geschäftsprozessen nicht mehr wegzudenken. Jeder Arbeitsschritt besitzt zeitliche Eigenschaften und meist gibt es zeitliche Bedingungen zwischen verschiedenen Vorgängen, deren Einhaltung überwacht und kontrolliert werden muss. Ein Prozess-Management-System, welches die Zeit ausreichend unterstützt, bringt daher viele Vorteile:

- Prozessschemata und im Speziellen deren Zeitbedingungen könnten bereits während der Modellierung auf Inkonsistenzen überprüft werden. Von einer Inkonsistenz spricht man, wenn eine der Zeitbedingungen aufgrund der anderen Zeiteigenschaften nicht eingehalten werden kann.

- Die Einhaltung der Zeitbedingungen kann zur Laufzeit der Prozessinstanzen überwacht und im Fall eines nicht Beachtens können spezielle Korrekturmaßnahmen (Eskalation) eingeleitet werden.

- Endanwender, welche durch das System unterstützt werden sollen, können über die aktuellen Zeitbedingungen ihrer Aufgaben informiert werden. Zusätzlich können ihnen Ablauf- und Termin-Pläne vorgeschlagen werden, welche eine Einhaltung der Zeitbedingungen garantieren (Ressourcenplanung).

- Aufgrund der gewonnenen Zeitdaten entstehen weitere Möglichkeiten zur Optimierung der Prozesse.

Diese Punkte tragen alle sowohl zu einer weiteren Entlastung der Endbenutzer als auch zu einer Steigerung der Effizienz bei und ermöglichen damit zusätzliche Kostenersparnisse.

Eine Unterstützung der Modellierung von Zeiteigenschaften ist allerdings auch mit einigen Problemstellungen verbunden. So unterliegen die Zeiteigenschaften der Prozesse und ihrer Teilaufgaben häufig größeren Schwankungen aufgrund äußerer Umstände. Auch

sind häufig wichtige Informationen für die Modellierung der Zeiteigenschaften während der Modellierungsphase nicht bekannt und können erst zur Laufzeit ermittelt werden. Außerdem muss das Zeitmanagement mit unvorhergesehenen Ereignissen wie dynamischen Änderungen oder einer Nichteinhaltung der Zeitbedingungen zurechtkommen. Bei einer Nichteinhaltung der Zeitbedingungen muss das Zeitmanagement den Benutzer auf diese aufmerksam machen, ihm aber auch gestatten, explizit von den gegebenen Zeitbedingungen abzuweichen. Dabei ist eine weitergehende Unterstützung des Benutzers durch das System notwendig, da die Folgen des Nichteinhaltens einer Zeitbedingung oft nur schwer abzuschätzen sind.

Wie man sieht, ist Zeit für Prozess-Management-Systeme ein wichtiges Thema, welches weit über die reine Verwaltung von Zeitdaten hinausgeht. Ziel dieser Arbeit ist es deshalb, neben der reinen Verwaltung von Zeitdaten insbesondere die genannten Probleme zu diskutieren und Lösungen aufzuzeigen. Hierzu werden zunächst die nötigen Anforderungen an das Zeitmanagement eines adaptiven PMS vorgestellt. Grundlage hierfür bildet das ADEPT2-System[1], welches maßgeblich an der Universität Ulm am Institut für Datenbanken und Informationssysteme entworfen wurde [Reichert & Dadam, 1997], [Dadam & Reichert, 1998]. Basierend auf ADEPT2 werden Verfahrensweisen für ein Zeitmanagement-Rahmenwerk, mit dem Ziel einer Integration in ein reales PMS wie ADEPT2, erarbeitet und die daraus resultierenden neuen Möglichkeiten erörtert.

## 1.1 Allgemeine Anforderungen

Ziel eines Prozess-Management-Systems ist vor allem eine Entlastung der Anwender. Dieses Ziel gilt daher auch für das Zeitmanagement. Es soll den Anwender von der Last der Planung und Koordination befreien und - daraus folgend - eine Reduzierung von Fehlern wie Fristverletzungen und Terminüberschreitungen und eine Unterstützung bei der Ressourcenverwaltung ermöglichen. Wichtig ist dabei vor allem eine intelligente Gestaltung der Benutzerschnittstellen.

Einerseits soll sie, neben ihrer eigentlichen Funktionalität, den Benutzer dazu bewegen, dringliche Aufgaben bevorzugt zu bearbeiten, andererseits ist es weder die Aufgabe der Benutzerschnittstellen noch des Zeitmanagements selbst, den Benutzer zur Einhaltung der Zeitrestriktionen zu zwingen. Sowohl das Zeitmanagement als auch die Benutzerschnittstelle sollten daher als ein intelligentes Assistenzsystem gestaltet werden, welches den Benutzer zwar unterstützt, ihn aber nicht bevormundet. Dazu muss es in der Lage sein, auf eine drohende oder eingetretene Verletzung der festgelegten Zeitkriterien zu reagieren. Hierzu zählt beispielsweise ein Notifikations- und Eskalations-Mechanismus, der den Benutzer zunächst auf anstehende Aufgaben aufmerksam macht, es dann ermöglicht, ihn auf drohende Verletzungen hinzuweisen, und schließlich auch eine Möglichkeit zur flexiblen Reaktion von Benutzer und System auf eingetretene Verletzungen ermöglicht.

[1] ADEPT steht für **A**pplication **D**evelopment Based on **E**ncapsulated Pre-Modeled **P**rocess **T**emplates

Die Anwendungsschnittstelle wiederum muss verschiedenste Möglichkeiten zur Beschaffung der Zeitdaten bieten, da diese aus den verschiedensten Quellen stammen können, wie:

- andere Zeitmanagement-Systeme (z. B. ein Terminplanungssystem)
- Benutzereingaben (z. B. in einem anderen Prozessschritt)
- Festlegungen auf Modellierungsebene (z. B. Geschäftsregeln oder Gesetze)
- vom System berechnet (d. h. aus anderen Zeitdaten abgeleitet)

Darüber hinaus muss das System so flexibel sein, dass einmal festgelegte Zeitdaten auch noch während der Laufzeit geändert werden können, um beispielsweise auf besondere Situationen reagieren zu können.

Ein letzter wichtiger Punkt ist, dass das Zeitmanagement in der Lage sein muss, aus den gegebenen Zeitdaten, -restriktionen und Terminen mögliche Ausführungszeitpunkte für die Aktivitäten abzuleiten. Dies ermöglicht eine Erstellung von Ablauf- und Terminplänen für die Benutzer und erlaubt in einem nächsten Schritt sogar die Anbindung eines Ressourcenmanagements an das PMS. Dies erweist sich sicherlich vor allem im Bereich von Prozess-Management-Systemen mit vielen gleichzeitig ablaufenden Prozessen als nützlich.

## 1.2 Einordnung und Aufgabenstellung der Diplomarbeit

Ein Projekt wie ADEPT2[2] und sein Vorgänger ADEPT [Hensinger *et al.*, 2000] umfasst eine große Anzahl unterschiedlicher Problemstellungen. Zunächst müssen ein adäquates Prozess-Metamodell für die Beschreibung der Prozesse entworfen und Kriterien für deren Korrektheit festgelegt werden. Es müssen Methoden, Vorgehensweisen und Regeln für die dynamische Modifizierbarkeit der Prozessinstanzen entwickelt werden. Benutzerschnittstellen und Interaktionsmöglichkeiten müssen ausgearbeitet werden. Organisatorische Aspekte müssen beachtet werden und vieles mehr. Zu ADEPT bzw. ADEPT2 gibt es daher bereits eine ganze Reihe von Arbeiten, die sich mit diesen und anderen Problemstellungen beschäftigen (z. B. [Hensinger, 1997], [Reichert & Dadam, 1998], [Rinderle *et al.*, 2003], [Rinderle *et al.*, 2004a], [Rinderle *et al.*, 2006], [Weber *et al.*, 2007], [Reichert & Bauer, 2007], [Ly *et al.*, 2008], [Dadam *et al.*, 2008]). ADEPT wurde dabei zunächst entwickelt, um erste Erfahrungen auf dem Gebiet der adaptiven Prozess-Management-Systeme zu sammeln. Als Fortsetzung hiervon soll ADEPT2 den ursprünglich entwickelten ADEPT-Prototypen auf eine stabile Basis stellen und ihn erweiterbar gestalten.

Die Aufgabe der vorliegenden Diplomarbeit ist es, die Anforderungen an eine Zeitplanungs- bzw. Zeitmanagementkomponente eines adaptiven Prozess-Management-Systems

[2] http://www.aristaflow.de Stand: 03.08.2008

anhand von ADEPT2 zu erarbeiten und zu beschreiben. Hierzu muss sowohl ein entsprechendes System hinsichtlich seiner Anforderungen an eine solche Komponente untersucht werden als auch ermittelt werden, welche Elemente zur Modellierung der zeitlichen Ebene eines Prozesses notwendig sind. Darauf aufbauend sind verschiedene Arbeiten zu Zeitaspekten und angrenzenden Themenbereichen auf ihre Eignung hin zu untersuchen. Anschließend sollen entsprechende Techniken für ein Rahmenwerk für das Zeitmanagement entworfen werden, wozu Algorithmen zur Konsistenzprüfung der definierten Zeitbedingungen der Prozessschemata und zur Planung von Terminen für die Aktivitäten entwickelt werden müssen. Außerdem ist zu untersuchen, welche neuen Aufgaben sich durch das Zeitmanagement für ein Prozess-Management-System ergeben und wie diese behandelt werden können. Das entwickelte Rahmenwerk für das Zeitmanagement soll anschließend prototypisch implementiert werden, um seine Anwendbarkeit zu demonstrieren.

Damit bildet die Arbeit die Grundlage für eine ausführliche Betrachtung weitergehender Aspekte wie beispielsweise eine intelligente Eskalations-Behandlung oder die Anbindung an ein System zum *Ressourcenmanagement*, welche in dieser Arbeit nur am Rande behandelt werden können. Ebenso nicht betrachtet wird eine Auswertung der Zeitdaten bereits abgeschlossener Prozesse und andere Techniken, welche beispielsweise im Rahmen des *Prozess-Reengineering* zur Anwendung kommen [van der Aalst & Weijters, 2005].

## 1.3 Aufbau der Arbeit

Die Arbeit ist wie folgt gegliedert: Nach der Einleitung werden in Kapitel 2 die Anforderungen an eine Zeitmanagementkomponente zunächst aus Sicht eines Prozess-Metamodells am Beispiel von ADEPT2 erörtert. Danach werden die für die Modellierung der Zeiteigenschaften nötigen Konstrukte und Eigenschaften ermittelt und deren Bedeutung sowie damit verbundene Probleme diskutiert.

In Kapitel 3 werden relevante Arbeiten zu Zeitaspekten vorgestellt und basierend auf den gewonnenen Erkenntnissen diskutiert. Neben unterschiedlichen Bereichen der Informatik sind dabei auch Arbeiten aus anderen Gebieten, wie beispielsweise der Projektplanung, vertreten.

Kapitel 4 stellt zunächst die für eine Modellierung der Zeiteigenschaften nötigen Veränderungen am Prozess-Metamodell des ADEPT2-Systems und die dabei auftretenden Probleme vor. In einem weiteren Schritt wird beschrieben, wie die Korrektheit der Modellierung überprüft und mögliche Ausführungszeitpunkte für die Aktivitäten berechnet werden können.

Bei der Anreicherung eines Prozess-Metamodells um Zeiteigenschaften gibt es einige Besonderheiten zu beachten, wie zum Beispiel Abhängigkeiten zwischen den Dauern der Aktivitäten und Abständen zwischen diesen, welche in Kapitel 5 anhand eines Beispiel-Prozesses diskutiert werden.

Nachdem ein Prozess vollständig spezifiziert wurde, kann er zur Ausführung gebracht werden. Welche Abläufe hierbei für das Zeitmanagement nötig sind und welchen Nutzen man während der Laufzeit aus diesem ziehen kann, legt Kapitel 6 dar. Außerdem wird hier diskutiert, wie das Zeitmanagement in die Laufzeit-Umgebung eines Prozess-Management-Systems wie ADEPT2 integriert werden kann.

Die wichtigsten der vorgeschlagenen Konzepte wurden im Rahmen der Arbeit prototypisch implementiert. Einige ausgewählte Aspekte dieser Implementierung werden in Kapitel 7 erörtert.

Die Arbeit schließt mit einer Zusammenfassung der wichtigsten Ergebnisse und einem Ausblick auf zukünftige Entwicklungen und Möglichkeiten in Kapitel 8.

# 2 Anforderungen an das Zeitmanagement

Die Anforderungen an das Zeitmanagement hängen maßgeblich von den Konstrukten des Prozess-Metamodells ab. Deshalb beginnt dieses Kapitel mit einer Vorstellung der üblichen Konstrukte eines solchen Prozess-Metamodells und der Untersuchung auf ihre Eigenschaften hinsichtlich der Zeit. Nützlich ist an dieser Stelle auch eine formale Definition dieser Konstrukte, wodurch deren Semantik eindeutig festgelegt wird. Schließlich muss ermittelt werden, welche Stadien ein Prozess während seiner Ausführung durchläuft, um später bestimmen zu können, wie das Zeitmanagement diese behandeln muss. All dies erfolgt in Abschnitt 2.1.

In Abschnitt 2.2 wird anschließend ermittelt, welche neuen Elemente dem Prozess-Metamodell hinzugefügt werden müssen, um die Modellierung der Zeiteigenschaften und -bedingungen zu ermöglichen. An dieser Stelle werden auch die verschiedenen Semantiken erläutert und bewertet, die für diese Zeitelemente gelten können. Abschließend wird noch auf das Thema der unterschiedlichen Zeiteinheiten eingegangen, in denen die Zeiteigenschaften angegeben werden können, und die in diesem Zusammenhang auftretenden Problemstellungen erläutert.

## 2.1 Prozess-Modellierung

In der Literatur ([van der Aalst *et al.*, 1994], [Reichert & Dadam, 1997], [Ellis & Nutt, 1993], [Attie *et al.*, 1993], [Wodtke & Weikum, 1997], etc.) finden sich verschiedene Metamodelle, die zur Modellierung eines Prozesses herangezogen werden können. Diese unterscheiden sich in ihrer Ausdrucksmächtigkeit, Komplexität, Semantik und noch weiteren Aspekten. Die meisten dieser Modelle definieren ein sogenanntes Basismodell, welches die grundlegenden Konstrukte des entsprechenden Modells enthält. Mithilfe dieser Basiskonstrukte werden dann die im Modell vorhandenen komplexeren Konstrukte zusammengesetzt. Der Vorteil eines Basismodells ist, dass es aufgrund der eng beschränkten Anzahl an Konstrukten erlaubt, ein gegebenes Prozessschema durch wenige und einfache Regeln auf Korrektheit zu überprüfen. Dadurch muss nicht auf komplexe Abhängigkeiten geachtet werden, was den nötigen Prüfaufwand erheblich senken kann.

Es gibt einige Konstrukte, die unter unterschiedlichen Namen und mit leicht veränderten Semantiken in fast allen (Basis-) Modellen zu finden sind. Dazu zählen beispielsweise Aktivitäten beziehungsweise Knoten, Kontrollkanten, Sequenzen, Parallelausführung und bedingte Auswahl.

Der für das Zeitmanagement relevante Teil dieser Konstrukte soll nun am Beispiel des *ADEPT2-Basismodells* [Reichert & Dadam, 1997], [Reichert & Dadam, 1998] vorgestellt und näher erläutert werden. Zugleich werden auch einige Konstrukte erörtert, welche nur im ADEPT2-Basismodell vorkommen, jedoch im Rahmen dieser Arbeit von Interesse sind.

Dabei wird, soweit möglich, auf die Terminologie der Workflow-Management-Coalition [WfMC, 1999] zurückgegriffen. Diese versucht einen gemeinsamen Standard bei den Prozess-Management-Systemen zu verbreiten, um deren Interoperabilität zu erhöhen. Dazu wurden unter anderem gemeinsame Termini und Semantiken festgelegt, die den Austausch unter den mit Prozess-Management beschäftigten Gruppen verbessern soll.

In Verbindung mit der Vorstellung der relevanten Konstrukte wird im Vorgriff auf Abschnitt 2.2 auch noch deren Bedeutung für das Zeitmanagement diskutiert werden.

Zunächst werden jedoch die Grundlagen von Prozess-Management-Systemen beschrieben, sowie einige für das Verständnis der nachfolgenden Abschnitte wichtige Begriffe definiert und genauer abgegrenzt.

### 2.1.1 Begriffsbildung

In diesem Abschnitt werden einige der bereits verwendeten oder im Nachfolgenden verwendeten Begriffe definiert und kurz erläutert. Dabei gilt das besondere Augenmerk den Begriffen, die im Zusammenhang mit Prozessen und Prozess-Management-Systemen auftreten. Begriffe in Zusammenhang mit Zeit und der Modellierung von Zeiteigenschaften und -bedingungen werden zu gegebener Zeit ebenfalls erläutert (siehe Abschnitt 2.2.1).

Einige der für die folgenden Begriffe verwendeten Definitionen sind aus [WfMC, 1999] entlehnt.

**Geschäftsprozess** Ein *Geschäftsprozess* ist eine Folge von Schritten, um ein Geschäftsziel zu erreichen. Geschäftsprozesse gehen oft über Abteilungen und Betriebsgrenzen hinweg und gehören zur Ablauforganisation eines Betriebs. Häufig sind die Ablaufstrukturen und Arbeitsschritte eines Geschäftsprozesses nicht explizit definiert, sondern ergeben sich durch die Erfahrung der Mitarbeiter und die Struktur des Unternehmens.

**Prozess** Ein *Prozess*, häufig auch als *Workflow* bezeichnet, ist eine formale Sicht eines Geschäftsprozesses, welche durch eine koordinierte Menge von Prozessaktivitäten repräsentiert wird, die so verbunden sind, dass ein bestimmtes, gemeinsames Ziel erreicht wird. Durch einen Prozess werden die Arbeitsschritte und Ablaufstrukturen eines Geschäftsprozesses formal definiert und damit explizit festgelegt. Die Darstellung eines einfachen Prozesses ist in Abbildung 2.1 zu sehen.

**Prozessschema** Die Darstellung eines Prozesses in einer Form, welche eine automatische Handhabung, wie beispielsweise Modellierung oder Ausführung

durch ein Prozess-Management-System erlaubt, wird als *Prozessschema* oder *Prozessmodell* bezeichnet. Ein Prozessschema besteht meist aus einem Netz von Aktivitäten und deren Beziehungen sowie weitergehenden Informationen über die einzelnen Aktivitäten.

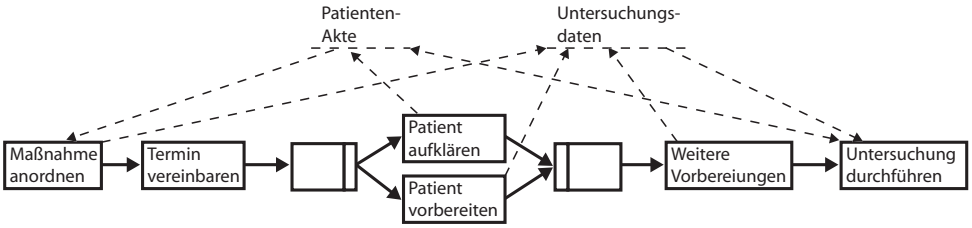



Abbildung 2.1: Teil einer medizinischen Untersuchung modelliert als Prozess

**Prozess-Management-System** Ein *Prozess-Management-System* (kurz: *PMS*) ist ein System, dass die Ausführung von Prozessen definiert, erzeugt und überwacht. Es verwendet dazu Software, welche in der Lage ist ein Prozessschema zu verstehen, mit den Teilnehmern zu interagieren und, wo nötig, externe Anwendungen aufzurufen.

**Prozess-Metamodell** Ein *Prozess-Metamodell* spezifiziert die Menge von Konstrukten, welche in einem PMS zur Modellierung eines Prozessschemas verwendet werden können.

**Basismodell** Ein *Basismodell* ist der einem Prozess-Metamodell zugrunde liegende, begrenzte Satz an Konstrukten und Operationen. Durch diesen werden die, auf Basis des Prozess-Metamodells, spezifizierten Prozessschemata überprüf- und einfach ausführbar.

**Kontrollfluss** Der *Kontrollfluss* beschreibt die zeitliche Abfolge der einzelnen Aktivitäten eines Prozesses. Dabei erlauben verschiedene Kontrollstrukturen, von der sequenziellen Abfolge der Aktivitäten abzuweichen.

**Prozessgraph** Ein *Prozessgraph* ist eine graphische Darstellung des Kontrollflusses eines Prozesses (siehe Abbildung 2.1), vergleichbar mit einem Aktivitätsdiagramm [Kecher, 2006]. Dabei kommen verschiedene Symbole zur Darstellung der unterschiedlichen Kontrollstrukturen zum Einsatz.

**Datenfluss** Der *Datenfluss* beschreibt, den Austausch der Prozessdaten zwischen verschiedenen Aktivitäten (siehe Abbildung 2.1).

**Aktivität** Eine *Aktivität* beschreibt ein Stück Arbeit, welche einen logischen Schritt innerhalb eines Prozesses bildet. Aktivitäten können automatisch, vom System

oder einem Programm, oder manuell, von einem menschlichen Benutzer, ausgeführt werden.

**Aktivitätsschema** Ein *Aktivitätsschema* ist eine abstrakte Beschreibung einer Aktivität für die Prozessmodellierung. Dabei wird üblicherweise die für die Aktivität nötige Anwendung sowie deren Ein- und Ausgabeparameter spezifiziert.

**Instanz** Eine *Instanz* repräsentiert eine einzelne Ausführung eines Prozesses oder einer Aktivität innerhalb eines Prozesses. Jede Instanz hat ihren eigenen internen Zustand und enthält die zur Ausführung gehörigen Daten. Eine Instanz ist vergleichbar einer Objektinstanz in der objektorientierten Programmierung [Oestereich, 1997], entsprechend repräsentiert ein Prozess- oder Aktivitätsschema die zugehörige Klasse.

**Prozessinstanz** Eine *Prozessinstanz* bezeichnet die Ausführung eines einzelnen Prozesses. Wird häufig auch als *Workflowinstanz* bezeichnet.

**Instanztypen** Der auf der linken Seite von Abbildung 2.2 dargestellte Ausschnitt eines Prozesses beinhaltet eine sogenannte *Alternativ-* oder *Exklusiv-Oder-Verzweigung* (Knoten B und F), bei der in jeder Prozessinstanz nur genau einer der von Knoten B ausgehenden Pfade durchlaufen wird (entweder C und D oder E, aber nicht beide). Diese unterschiedlichen Varianten bei der Ausführung eines Prozesses werden als *Instanztypen* [Marjanovic & Orlowska, 1999] bezeichnet.

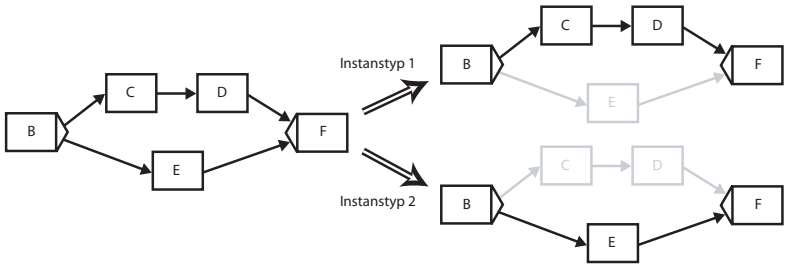



Abbildung 2.2: Instanztypen

**Aktivitätsinstanz** *Aktivitätsinstanzen* bezeichnen die Ausführung einer einzelnen Aktivität innerhalb einer Prozessinstanz.

**Agent** Ein *Agent* oder auch *Bearbeiter* ist eine menschliche oder maschinelle Ressource, welche die Arbeit, die durch eine Aktivitätsinstanz repräsentiert wird, durchführt. Diese Arbeit wird dem Agenten meist über seine Arbeitsliste zugewiesen.

**Arbeitsliste** Eine *Arbeitsliste* ist eine Menge von Arbeitsaufträgen, die einem Agenten oder einer Gruppe von Agenten zugewiesen wurde. Eine Arbeitsliste stellt somit

einen Teil der Schnittstelle zwischen den Agenten und dem Prozess-Management-System dar.

**zeitliche Unsicherheit** Als *zeitliche Unsicherheit* oder *temporale Unsicherheit* wird der Grad der Ungenauigkeit einer zeitlichen Aussage aufgrund unbekannter äußerer Umstände bezeichnet. Beispielsweise schwankt die Dauer einer Aktivität häufig zwischen verschiedenen Ausführungen. Dann ist es meist nur möglich, eine minimale und maximale Dauer für die Aktivität anzugeben. Die Differenz zwischen diesen beiden Dauern bestimmt dann die zeitliche Unsicherheit.

Nachdem die nötigen Begriffe erläutert wurden, werden diese nun bei der Vorstellung des ADEPT2-Basismodells verwendet.

### 2.1.2 ADEPT2-Basismodell

ADEPT2 verwendet für die Definition eines *Prozessschemas* einen graphischen, netzbasierten Ansatz [Reichert & Dadam, 1998], [Rinderle *et al.*, 2004c], [Rinderle *et al.*, 2004b]. Dieser wurde vor allem unter dem Aspekt der dynamischen Modifizierbarkeit entwickelt, weswegen sowohl Kontroll- als auch Datenfluss explizit modelliert werden. Dadurch ist es möglich, statische und dynamische Analysen zum Erhalt der semantischen Korrektheit durchzuführen [Hensinger, 1997], [Reichert & Dadam, 1998], [Reichert, 2000].

Grundlage für die Spezifikation des *Kontrollflusses* ist das Konzept der symmetrischen Kontrollstrukturen, auch *Blockstruktur* genannt [Reichert & Dadam, 1997]. Dieses ist rekursiv auf Basis von Blöcken definiert. Ein Block kann dabei sowohl einfach, als auch komplex sein. Einfache Blöcke sind atomar (d. h., sie können nicht weiter zerlegt werden), wohingegen komplexe Blöcke in weitere einfache oder komplexe Blöcke zerlegt werden können (siehe Abbildung 2.3). Ein Prozessschema ist zunächst ein komplexer Block, der in einen Start- und End-Knoten sowie einen in Sequenz dazwischen eingeordneten Block zerlegt werden kann. Einfache Blöcke repräsentieren Aktivitäten oder Subprozesse, die aus Sicht des Prozesses nicht weiter zerlegt werden können. Sie werden im Prozessgraphen durch einfache Knoten dargestellt. Komplexe Blöcke hingegen können in weitere einfache oder komplexe Blöcke zerlegt werden. Diese können dabei sequenziell oder parallel angeordnet sein oder die Sub-Blöcke werden in einer Schleife wiederholt ausgeführt. Die Zerlegung eines Blocks in parallele Blöcke oder in eine Schleife wird im Prozessschema durch spezielle Start- und End-Knoten gekennzeichnet (siehe Abbildung 2.4).

Bei ADEPT2 werden für jede Aktivität zugehörige Ein- und Ausgabeparameter festgelegt, mit deren Hilfe der Informationsfluss zwischen den Aktivitäten, über eine spezielle Datenfluss-Modellierung, beschrieben wird [Rinderle-Ma *et al.*, 2008]. Dies ist vor allem unter dem Gesichtspunkt der dynamischen Änderungen von Interesse [Reichert & Dadam, 1997], [Reichert & Dadam, 1998], da auf dieser Basis die Korrektheit des Datenflusses jederzeit gewährleistet werden kann. Modelliert wird der *Datenfluss* über sogenannte

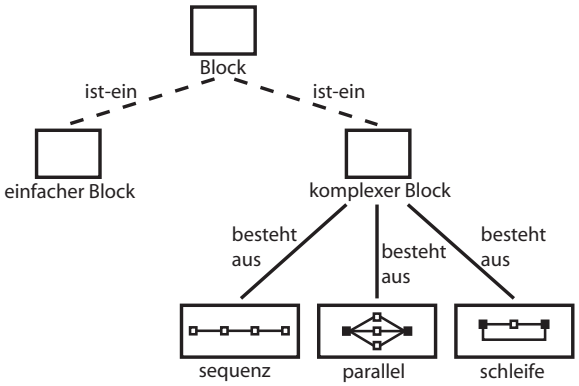



Abbildung 2.3: Einfache und komplexe Blöcke

*Datenelemente* und *Datenkanten*. Ein Datenelement dient zur Speicherung eines bestimmten Wertes. Datenkanten verbinden die Datenelemente mit den Aktivitäten. Dabei stellt eine von einer Aktivität ausgehende Datenkante einen schreibenden und eine eingehende Datenkante einen lesenden Zugriff auf den im Datenelement gespeicherten Wert dar. Für das Zeitmanagement spielt der Datenfluss keine Rolle, weswegen er im Folgenden nur am Rande Erwähnung finden wird.

Als Basis für die Integration eines Zeitmanagements ist ADEPT2 nach ersten Überlegungen gut geeignet. Die formale Basis und die verwendete Blockstruktur bieten ein solides Fundament, auf dem die einzuführenden Zeitkonstrukte syntaktisch und semantisch definiert werden können. Außerdem bietet ADEPT2 die nötigen Voraussetzungen, um zur Laufzeit dynamische Prozessänderungen einbringen und auf Konsistenz überprüfen zu können. Dies ist vor allem im Zusammenhang mit Abständen und Dauern, die erst während der Ausführung der *Prozessinstanzen* bekannt sind, von Interesse (siehe Abschnitt 2.2). Aufgrund der genannten Vorteile wird im Folgenden das ADEPT2-Metamodell als Grundlage verwendet. Die gewonnenen Erkenntnisse lassen sich jedoch auch auf andere Modelle übertragen.

Im Folgenden wird ein auf den für das Zeitmanagement nötigen Teil der Konstrukte reduziertes ADEPT2-Basismodell vorgestellt. Die Konstrukte werden zunächst informell eingeführt und in Abschnitt 2.1.3 formal definiert. Danach werden einige zur Laufzeit nötige Erweiterungen des Modells beschrieben. In Kapitel 4 wird das ADEPT2-Modell um einige für das Zeitmanagement nötige Aspekte erweitert.

#### 2.1.2.1 Knoten

*Knoten* repräsentieren einen, zumindest aus Sicht des Prozessgraphen, elementaren Arbeitsschritt. Es ist daher zunächst unerheblich, ob sich hinter einem Knoten eine einfache *Aktivität* oder ein komplexer *Sub-Prozess* verbirgt. Im Folgenden werden die

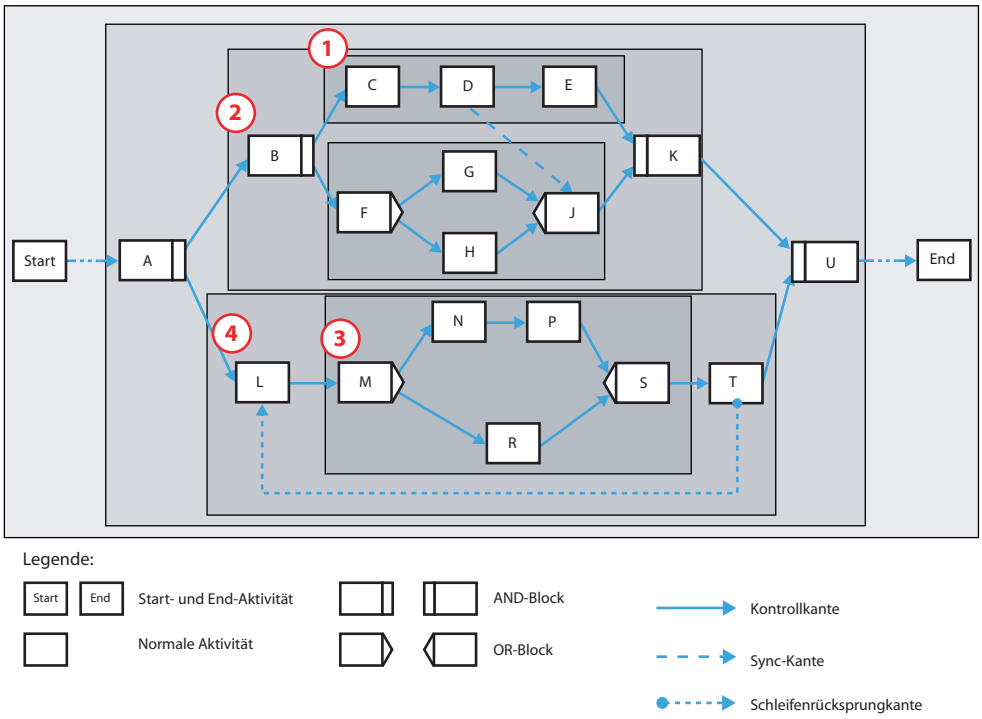



Abbildung 2.4: Darstellung eines Prozessschemas in ADEPT2 [Jurisch, 2006]

Begriffe Aktivität und Knoten daher häufig synonym verwendet. Neben den Knoten zur Repräsentation von Arbeitsschritten gibt es noch Spezial-Knoten, die lediglich dazu dienen, den Prozessgraphen zu strukturieren (siehe Abschnitte zu Parallel-Verzweigungen, Alternativ-Verzweigungen und Schleifen weiter unten).

Aktivitäten und auch Subprozesse benötigen Zeit, um die zugrundeliegende Aufgabe zu erledigen. Aus diesem Grund besitzt ein Knoten aus zeitlicher Sicht eine gewisse Dauer, die vom Betreten des Knotens, das heißt dem Starten seiner Aktivität, bis zu seinem Verlassen, das heißt dem Ende seiner Aktivität, vergeht. Diese Dauer kann Schwankungen unterliegen, was bei der Modellierung bzw. der Wahl des Modells beachtet werden muss.

#### 2.1.2.2 Kontrollkanten

*Kontrollkanten* bilden die Basis eines jeden Prozesses. Sie definieren die Nachfolgerrelation der Aktivitäten des Prozessschemas und geben damit den Kontrollfluss innerhalb des Prozessgraphen an. Sie werden zumeist als einfache Kanten oder Pfeile dargestellt, die zwei Aktivitäten (Vorgänger und Nachfolger) miteinander verbinden (siehe Abbildung 2.4).

Zeitlich betrachtet haben Kontrollkanten die Bedeutung, dass die Ziel-Aktivität der Kante frühestens gestartet werden kann, nachdem ihre Quell-Aktivität beendet wurde. Sie entsprechen daher einem positiven Minimal-Abstand zwischen dem Ende der Quell- und dem Start der Ziel-Aktivität.

#### 2.1.2.3 Sequenzen

Eine *Sequenz* stellt die einfachste Form eines Kontrollkonstrukts dar. Dabei werden zwei oder mehr Aktivitäten nacheinander ausgeführt. Jede der beteiligten Aktivitäten einer Sequenz besitzt somit genau einen Nachfolger, der nach dem Beenden der Aktivität aktiviert wird. Modelliert wird eine Sequenz zweier Aktivitäten dadurch, dass die beiden Aktivitäten durch eine Kontrollkante verbunden werden. Weitere Aktivitäten werden der Sequenz hinzugefügt, in dem diese an den Start oder das Ende der Sequenz durch eine weitere Kontrollkante angefügt werden (siehe ① in Abbildung 2.4). Alle Aktivitäten einer Sequenz haben daher genau eine eingehende und eine ausgehende Kontrollkante.

Für die Zeitberechnung stellt die Sequenz das einfachste Konstrukt dar. Sofern die Dauern der beteiligten Aktivitäten bekannt sind, können genaue Vorhersagen über die frühesten Ausführungszeitpunkte der Aktivitäten und ihrer Nachfolger getroffen werden. Um über die spätesten Ausführungszeitpunkte ebenfalls eine Vorhersage abgeben zu können, muss zwischen der ersten und letzten Aktivität der Sequenz zusätzlich noch ein (möglicherweise impliziter) Maximal-Abstand existieren, welcher die Zeit einschränkt, die zwischen dem Start der ersten und dem Ende der letzten Aktivitäten vergehen darf. Eventuelle Unsicherheiten bei den Dauern der einzelnen Aktivitäten, zum Beispiel durch unterschiedliche Minimal- und Maximal-Dauern, addieren sich über die Aktivitäten einer Sequenz hinweg auf. Es entstehen durch die Sequenz jedoch keine weiteren *zeitlichen Unsicherheiten*.

#### 2.1.2.4 Parallel-Verzweigungen

*Parallel-Verzweigungen*, auch *AND-Blöcke* genannt, stellen ein paralleles Kontrollkonstrukt dar. Sie bestehen aus einem *AND-Split / AND-Join* Doppelkonstrukt (siehe ② in Abbildung 2.4). Am AND-Split teilt sich der eingehende Kontrollfluss in zwei oder mehr parallele Kontrollflüsse auf, die dann voneinander unabhängig ablaufen. Dabei ist als Besonderheit zu beachten, dass die Aktivitäten der verschiedenen Kontrollflüsse gleichzeitig ablaufen können, dies aber nicht müssen. Es ist auch möglich, dass die Aktivitäten in beliebiger Reihenfolge, wie bei einer Sequenz, nacheinander ausgeführt werden. Sämtliche aus einem AND-Split ausgehenden Kontrollflüsse müssen schlussendlich durch einen AND-Join wieder in einen gemeinsamen Kontrollfluss vereint werden. Der AND-Join bzw. dessen Nachfolger wird erst dann aktiviert, wenn alle Aktivitäten der eingehenden Kontrollflüsse beendet sind. Ein AND-Split hat somit zwei oder mehr ausgehende Kontrollkanten und der entsprechende AND-Join genau so viel eingehende Kanten. Zwischen

den Aktivitäten verschiedener paralleler Kontrollflüsse können keine Kontrollkanten existieren. Um die Ausführungsreihenfolge parallel angeordneter Aktivitäten dennoch beeinflussen zu können, existieren sogenannte Synchronisationskanten.

Für die Zeitberechnung stellen Parallel-Verzweigungen kein großes Problem dar. Die Dauer, die zwischen AND-Split und AND-Join vergeht, ist genau so lange, wie die maximale Dauer, die einer der dazwischen liegenden Pfade benötigt. Dabei addieren sich, wie bei der Sequenz, eventuelle *zeitliche Unsicherheiten* bei den Dauern lediglich über die Aktivitäten der dazwischenliegenden Pfade hinweg auf.

#### Synchronisationskanten

Manchmal ist es nötig, Abhängigkeiten in der Ausführungsreihenfolge der Aktivitäten parallel ablaufender Kontrollflüsse zu modellieren. Hierfür dienen sogenannte *Synchronisationskanten* (kurz: *Sync-Kanten*). Diese ermöglichen es, die Ausführungsreihenfolge der Aktivitäten paralleler Kontrollflüsse zu beeinflussen. Beispielsweise wird durch die Sync-Kante in Abbildung 2.4 Aktivität D mit Aktivität J synchronisiert. Dies bedeutet, dass Aktivität J erst ausgeführt werden kann, nachdem D erfolgreich ausgeführt wurde, oder wenn klar ist, dass D nicht zur Ausführung kommt. Die Semantik ist somit ähnlich, wie die einer Kontrollkante. Dabei ist zu beachten, dass eine Sync-Kante für die Sequenz-Eigenschaft einer Gruppe von Aktivitäten keine Rolle spielt.

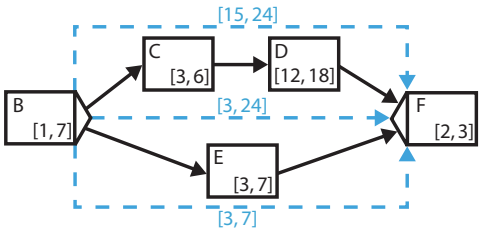
Sync-Kanten entsprechen in Bezug auf die Zeitrechnung ebenfalls einfachen Kontrollkanten und drücken damit einen positiven Minimal-Abstand aus. Jedoch ergeben sich weitere Erschwernisse aus dem Umstand, dass zum einen die parallelen Zweige des AND-Blocks nicht mehr in beliebiger Reihenfolge abgearbeitet werden können, und zum anderen, dass der Startzeitpunkt mancher Aktivitäten eventuell nach hinten verschoben werden muss, was im Zusammenhang mit Maximal-Abständen zu weiteren Problemen führen kann.

### 2.1.2.5 Alternativ-Verzweigungen

*Alternativ-Verzweigungen*, auch *XOR-Blöcke* genannt, repräsentieren alternative Ablaufpfade bei der Ausführung eines Prozesses. Sie werden wie Parallel-Verzweigungen durch ein Doppelkonstrukt, bestehend aus einem *XOR-Split* und einem zugehörigen *XOR-Join*, modelliert (siehe ③ in Abbildung 2.4). Dabei muss sich der Kontrollfluss, beim Erreichen des XOR-Splits für einen der ausgehenden Pfade entscheiden. Dieser wird weiter ausgeführt, wohingegen die Übrigen abgewählt werden und dementsprechend nicht mehr zur Ausführung kommen. Die abzuwählenden Pfade stehen dabei, aufgrund der gewählten Blockstrukturierung, eindeutig fest. Der XOR-Join dient dazu, die verschiedenen möglichen Ausführungspfade wieder auf einen gemeinsamen Pfad zu vereinigen. Ein XOR-Split hat somit, wie ein AND-Split, zwei oder mehr ausgehende Kontrollkanten und der XOR-Join entsprechend viele eingehende Kontrollkanten, von denen in jeder

Prozessinstanz jeweils nur genau eine aktiv ist. Das heißt, der XOR-Join kann ausgeführt werden, sobald einer der eingehenden Pfade abgeschlossen ist.

Für das Zeitmanagement stellt eine Alternativ-Verzweigung eine Herausforderung dar. Das Problem besteht darin, dass die einzelnen Pfade im Allgemeinen unterschiedliche Laufzeiten aufweisen. Daher können, solange keine Gewissheit herrscht, welche der Alternativen gewählt wird, die Zeitpunkte der nachfolgenden Aktivitäten nicht ohne Weiteres berechnet werden. Hinzu kommt, dass sich die *zeitliche Unsicherheit* des gesamten Konstruktes nicht aus der zeitlichen Unsicherheit eines Weges, sondern aus der Vereinigung der zeitlichen Unsicherheiten aller möglichen Wege ergibt (siehe Beispiel 2.1). Es gibt verschiedene Möglichkeiten mit diesem Problem umzugehen, wie zum Beispiel die Beschränkung auf den kürzesten und längsten Pfad oder das Auffalten des Prozessgraphen. Diese werden in Abschnitt 4.1.6 näher betrachtet sowie ihre Vor- und Nachteile eingehend diskutiert.




Der obere Pfad der dargestellten Alternativ-Verzweigung besitzt eine zeitliche Unsicherheit von $24 - 15 = 9$ und der untere eine zeitliche Unsicherheit von $7 - 3 = 4$. Die gesamte Alternativ-Verzweigung hingegen besitzt eine zeitliche Unsicherheit von $24 - 3 = 21$.

Beispiel 2.1: Zeitliche Unsicherheit einer Alternativ-Verzweigung

#### 2.1.2.6 Schleifen

Manchmal müssen Abschnitte des Prozesses wiederholt werden, wobei zur Modellierungszeit nicht unbedingt bekannt ist, wie oft dies geschehen muss. Um diesen Sachverhalt dennoch modellieren zu können, gibt es das *Schleifen*-Konstrukt. Bei einer Schleife handelt es sich um einen Block, eingeleitet durch einen *STARTLOOP*- und beendet durch einen *ENDLOOP*-Knoten, zwischen denen sich beliebige weitere Konstrukte beziehungsweise Blöcke befinden können (siehe ④ in Abbildung 2.4). Der ENDLOOP ist zusätzlich zu den beiden Kontrollkanten, die ihn mit seinem Vorgänger und seinem Nachfolger verbinden, noch durch eine spezielle Kante, eine sogenannte *Schleifenrückwärtskante*, mit dem entsprechenden STARTLOOP verbunden. Die Semantik einer Schleife ist die, dass die Schleife nach dem Aktivieren des STARTLOOP-Knotens zunächst einmal vom Kontrollfluss durchlaufen wird. Beim ENDLOOP angekommen wird dann entschieden,

ob ein Rücksprung entlang der Schleifenrückwärtskante zu erfolgen hat, das heißt eine Wiederholung des Schleifenblocks erfolgen soll, oder ob die Schleife verlassen wird.

Eine Schleife stellt für das Zeitmanagement das schwierigste Konstrukt dar, da zur Modellierzeit oftmals keine Aussage über die Anzahl der Iterationen getroffen werden kann und diese selbst zur Laufzeit häufig erst beim letzten Durchgang der Schleife feststeht. Kann eine maximale Anzahl an Iterationen der Schleife bestimmt werden, so kann die Schleife aus Sicht des Zeitmanagements als eine Kaskade von XOR-Konstrukten repräsentiert werden, wobei nach jeder Iteration durch ein XOR-Split entschieden wird, ob die restlichen Iterationen übersprungen oder eine weitere Iteration durchgeführt wird (siehe Abbildung 2.5). Da die minimale und maximale Anzahl an Iterationen aber möglicherweise sehr weit differieren, erhält man hier ein XOR-Konstrukt mit sehr vielen möglichen Zweigen, was eine genaue Zeitberechnung sehr erschwert. Auch die zeitliche Unsicherheit von Schleifen ist sehr groß, da die Zeitdauer der Schleife zwischen der minimalen Zeitdauer für einen Durchlauf und einer potenziell unendlichen maximalen Zeitdauer aller Durchläufe schwanken kann. Befinden sich innerhalb der Schleife zusätzlich noch ein oder mehrere XOR-Konstrukte oder sogar weitere Schleifen, erhöhen sich sowohl die Probleme für die Berechnung als auch die zeitliche Unsicherheit beträchtlich.

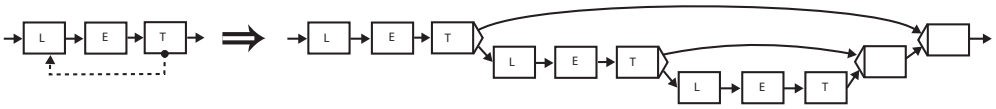



Abbildung 2.5: Darstellung einer Schleife als Kaskade von XOR-Konstrukten

### Schleifenrückwärtskanten

*Schleifenrückwärtskanten* können nur zwischen dem End- und dem Anfangsknoten einer Schleife (ENDLOOP bzw. STARTLOOP) vorkommen. Es handelt sich dabei um eine spezielle Kante, die dem ansonsten azyklischen Prozessgraphen die Möglichkeit zur Wiederholung einzelner Bereiche hinzufügt, ohne dabei die Azyklität des Prozessgraphen zu „zerstören“. Dies wird dadurch erreicht, dass beim Verfolgen einer Schleifenrückwärtskante der Schleifenrumpf wieder in den ursprünglichen Zustand zurückgesetzt wird.

Aus Sicht der Zeit betrachtet, handelt es sich bei Schleifenrückwärtskanten, wie bei den anderen Kanten, um einen positiven Minimal-Abstand zwischen ihrem Quell- und Zielknoten. Jedoch ist als Besonderheit zu beachten, dass der Zielknoten und dessen Nachfolger bereits einen Zeitpunkt zugewiesen bekommen haben, der einige Zeit vor dem neuen Zeitpunkt liegt. Daher können Schleifenrückwärtskanten nicht einfach als Minimal-Abstand in die Zeitrechnung übernommen werden.

### 2.1.3 Formale Betrachtung

Um eine präzise Semantik zu erhalten, definieren wir nun die zuvor informell vorgestellten Konstrukte formal. Dies erlaubt eine einfache Verwendung der Konstrukte und ihrer Eigenschaften in den verschiedenen Formeln und Algorithmen, die im Laufe dieser Arbeit vorgestellt werden.

#### 2.1.3.1 Prozesse

Ein *Prozess* im ADEPT2-Basismodell kann durch einen gerichteten Graphen

$$P = (\mathbf{N}, \mathbf{E}, \mathbf{D}, \mathbf{DF})$$

repräsentiert werden. Dabei ist $\mathbf{N}$ eine endliche Menge von *Knoten*[1] und $\mathbf{E}$ eine endliche Menge von *Kanten*[2]. $\mathbf{D}$ und $\mathbf{DF}$ dienen zur Beschreibung des Datenflusses. Bei $\mathbf{D}$ handelt es sich um eine Menge von *Datenelementen* und bei $\mathbf{DF}$ um eine Menge von *Datenkanten*. Da die Modellierung und Überprüfung des Datenflusses für das Zeitmanagement nicht von Bedeutung ist, wird im Rahmen dieser Arbeit nicht weiter auf diese beiden Elemente eingegangen werden. Der interessierte Leser sei an dieser Stelle auf [Reichert, 2000, S. 114ff.] verwiesen.

#### 2.1.3.2 Knoten

Ein *Knoten* $n \in \mathbf{N}$ wird durch ein 4-Tupel

$$(\mathtt{ID}_n, \mathtt{nT}_n, \mathbf{Pars}_n, \mathbf{M}_n)$$

repräsentiert. Dabei ist $\mathtt{ID}_n$ ein innerhalb des Prozesses eindeutiger Bezeichner für den Knoten, mit dem dieser identifiziert werden kann. $\mathtt{nT}_n \in \mathbf{nT}$ bestimmt den Typ des Knotens, wobei $\mathbf{nT}$ die Menge der für einen Knoten möglichen *Typen* bezeichnet:

- STARTWF/ENDWF für Start- und End-Knoten des Prozessschemas. Die Menge der Knoten $\mathbf{N}$ enthält jeweils genau einen Knoten des Typs STARTWF und einen des Typs ENDWF.
- ACTIVITY für Knoten die eine normale *Aktivität* oder einen *Sub-Prozess* enthalten.
- AND–SPLIT/-JOIN für *AND-Split* und *AND-Join* Knoten.
- XOR–SPLIT/-JOIN für *XOR-Split* und *XOR-Join* Knoten.
- START-/ENDLOOP für Anfangs- und End-Knoten einer *Schleife*.

[1] Englisch: ***N**ode*
[2] Englisch: ***E**dge*

Bei $\mathbf{Pars}_n$ handelt es sich um Parameter für die Datenversorgung der Aktivitäten eines Prozesses. Wie eingangs erwähnt, wird hierauf nicht näher eingegangen. $\mathbf{M}_n$ umfasst eine Menge von *Meta-Daten* die bei jedem Knoten hinterlegt werden können. Jedes Meta-Datum $m \in \mathbf{M}_n$ besteht aus einem Tupel $(\mathtt{mID}, value)$; $\mathtt{mID}$ ist ein eindeutiger Bezeichner für das Meta-Datum und *value* ist ein beliebiger Wert, dessen Typ von der Art des Meta-Datums abhängt. Meta-Daten können beispielsweise dazu verwendet werden, die *Entscheidungsparameter* eines Knotens vom Typ XOR–Split oder ENDLOOP zu hinterlegen. Die Entscheidungsparameter geben an, unter welchen Bedingungen ein bestimmter Pfad eines XOR-Splits gewählt bzw. eine Schleife wiederholt werden soll.

#### 2.1.3.3 Kanten

Eine *Kante* $e \in \mathbf{E}$ ist als Tripel

$$(\mathtt{ID}_e^{start}, \mathtt{ID}_e^{dest}, \mathtt{eT}_e)$$

definiert. Dabei identifizieren $\mathtt{ID}_e^{start}$ und $\mathtt{ID}_e^{dest}$ Start- und Zielknoten der Kante, wobei $\mathtt{ID}_e^{start} \neq \mathtt{ID}_e^{dest}$ gelten muss. Der *Kantentyp* wird durch $\mathtt{eT}_e \in \mathbf{eT}$ festgelegt. Die möglichen Werte von $\mathbf{eT}$ sind

- CONTROL_EDGE für eine normale *Kontrollkante*,
- SYNC_EDGE für eine *Synchronisationskante* und
- LOOP_EDGE für eine *Schleifenrückwärtskante*.

#### 2.1.3.4 Weitere Konventionen

Mithilfe der Knoten- und Kantentypen lassen sich weitere nützliche Mengen von Knoten und Kanten definieren. Wichtige Knotenmengen sind:

- Die Menge der Knoten mit Aktivitäten:

$$Activity\_n = \{n \in \mathbf{N} | \mathtt{nT}_n = \text{ACTIVITY}\}$$

- Die Menge der Split-Knoten:

$$Split\_n = \{n \in \mathbf{N} | \mathtt{nT}_n = \text{AND–SPLIT} \vee \mathtt{nT}_n = \text{XOR–SPLIT}\}$$

- Die Menge der Join-Knoten:

$$Join\_n = \{n \in \mathbf{N} | \mathtt{nT}_n = \text{AND–JOIN} \vee \mathtt{nT}_n = \text{XOR–JOIN}\}$$

Relevante Kantenmengen sind:

- Die Menge der Kontrollkanten:

$$Control\_e = \{e \in \mathbf{E} | \mathtt{eT}_e = \text{CONTROL\_EDGE}\}$$

- Die Menge der Sync-Kanten:

$$Sync\_e = \{e \in \mathbf{E} | \mathtt{eT}_e = \text{SYNC\_EDGE}\}$$

- Die Menge der Schleifenrückwärtskanten:

$$Loop\_e = \{e \in \mathbf{E} | \mathtt{eT}_e = \text{LOOP\_EDGE}\}$$

Die mögliche Anzahl der von einem Knoten ein- und ausgehenden Kanten hängt vom Typ des jeweiligen Knotens und dem Typ der Kanten ab. Mit den eben definierten Mengen lassen sich die wichtigsten Bedingungen angeben:

- Der Startknoten eines Prozesses besitzt keine eingehenden Kanten:

$$\begin{aligned}&\forall(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in \mathbf{E}, (\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in \mathbf{N}:\\ &(\mathtt{ID}_e^{dest} = \mathtt{ID}_n \Rightarrow \mathtt{nT}_n \neq \text{STARTWF})\end{aligned}$$

- Der End-Knoten eines Prozesses besitzt keine ausgehenden Kanten:

$$\begin{aligned}&\forall(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in \mathbf{E}, (\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in \mathbf{N}:\\ &(\mathtt{ID}_e^{start} = \mathtt{ID}_n \Rightarrow \mathtt{nT}_n \neq \text{ENDWF})\end{aligned}$$

- Jede Aktivität besitzt genau eine ausgehende und eine eingehend Kontrollkante:

$$\begin{aligned}&\forall(\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in Activity\_n:\\ &\quad|\{(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Control\_e | \mathtt{ID}_e^{start} = \mathtt{ID}_n\}| = 1\\ &\quad\wedge|\{(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Control\_e | \mathtt{ID}_e^{dest} = \mathtt{ID}_n\}| = 1\end{aligned}$$

- Eine Schleifenrückwärtskante verbindet immer einen Knoten vom Typ ENDLOOP mit einem Knoten vom Typ STARTLOOP und jeder ENDLOOP wird durch eine Schleifenrückwärtskante mit einem STARTLOOP verbunden:

$$\begin{aligned}&\forall(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Loop\_e:\\ &\quad(\forall(\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in \mathbf{N} : \mathtt{ID}_e^{start} = \mathtt{ID}_n \Rightarrow \mathtt{nT}_n = \text{ENDLOOP})\\ &\quad\wedge(\forall(\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in \mathbf{N} : \mathtt{ID}_e^{dest} = \mathtt{ID}_n \Rightarrow \mathtt{nT}_n = \text{STARTLOOP})\end{aligned}$$

und

$$\begin{aligned}&\forall(\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in \mathbf{N}:\\ &\quad(\mathtt{nT}_n = \text{ENDLOOP} \Rightarrow \exists(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Loop\_e : \mathtt{ID}_n = \mathtt{ID}_e^{start})\end{aligned}$$

- Jeder Split-Knoten hat genau eine eingehende und mindestens eine ausgehende Kontrollkante:

$$\begin{aligned}&\forall(\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in Split\_n :\\ &\quad(|\{(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Control\_e | \mathtt{ID}_e^{dest} = \mathtt{ID}_n\}| = 1\\ &\quad\wedge |\{(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Control\_e | \mathtt{ID}_e^{start} = \mathtt{ID}_n\}| \geq 1)\end{aligned}$$

- Analog hat jeder Join-Knoten mindestens eine eingehende und genau eine ausgehende Kontrollkante:

$$\begin{aligned}&\forall(\mathtt{ID}_n,\mathtt{nT}_n,\mathbf{Pars}_n,\mathbf{M}_n) \in Join\_n :\\ &\quad(|\{(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Control\_e | \mathtt{ID}_e^{dest} = \mathtt{ID}_n\}| \geq 1\\ &\quad\wedge |\{(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in Control\_e | \mathtt{ID}_e^{start} = \mathtt{ID}_n\}| = 1)\end{aligned}$$

#### 2.1.3.5 Nachbarschaftsfunktionen

Eine wichtige Rolle für die Zeitplanung spielen die Vorgänger- und Nachfolgerbeziehungen eines Knotens innerhalb des Prozessgraphen. Diese sind wie folgt definiert:

**Definition 2.1** (Nachfolger)**.** Die Nachfolgerrelation $\succ$ bestimmt die **direkten** Nachfolger eines Knotens über einer Kantenmenge:
$\succ \subset \mathbf{N} \times \mathbf{N}$ mit

$$n' \succ_E n \Leftrightarrow \exists(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in E : \mathtt{ID}_e^{start} = \mathtt{ID}_n \wedge \mathtt{ID}_e^{dest} = \mathtt{ID}_{n'}$$

Die Relation $\succcurlyeq$ bestimmt die transitive Hülle der Nachfolgerrelation:
$\succcurlyeq \subset \mathbf{N} \times \mathbf{N}$ mit

$$n' \succcurlyeq_E n \Leftrightarrow (n' \succ_E n) \vee (\exists n'' \in \mathbf{N} : n' \succ_E n'' \wedge n'' \succcurlyeq_E n)$$

**Definition 2.2** (Vorgänger)**.** Die Vorgängerrelation $\prec$ bestimmt die **direkten** Vorgänger eines Knotens über einer Kantenmenge:
$\prec \subset \mathbf{N} \times \mathbf{N}$ mit

$$n' \prec_E n \Leftrightarrow \exists(\mathtt{ID}_e^{start},\mathtt{ID}_e^{dest},\mathtt{eT}_e) \in E : \mathtt{ID}_e^{dest} = \mathtt{ID}_n \wedge \mathtt{ID}_e^{start} = \mathtt{ID}_{n'}$$

Die Relation $\preccurlyeq$ bestimmt die transitive Hülle der Vorgängerrelation:
$\preccurlyeq \subset \mathbf{N} \times \mathbf{N}$ mit

$$n' \preccurlyeq_E n \Leftrightarrow (n' \prec_E n) \vee (\exists n'' \in \mathbf{N} : n' \prec_E n'' \wedge n'' \preccurlyeq_E n)$$

### 2.1.4 Prozessausführung

Wird ein *Prozessschema* zur Ausführung gebracht, so wird von diesem zunächst eine *Prozessinstanz* erzeugt. Hierfür müssen die Knoten und Kanten des Prozessschemas um instanzspezifische Zustandsdaten angereichert werden, die den aktuellen Ausführungszustand der einzelnen Elemente beschreiben.

Der Ausführungszustand eines Knotens setzt sich aus seinem aktuellen Knotenzustand und der Anzahl seiner bisherigen Durchführungen zusammen, d. h., die Knotenbeschreibung wird um die beiden Elemente $\texttt{It}_n \in \mathbb{N}_0$ für die Anzahl der bisher durchgeführten Iterationen sowie $\texttt{Ns}_n \in \mathbf{NS}$ für den aktuellen Knotenzustand erweitert. Die Anzahl der bisher durchgeführten Iterationen ist in verschiedenen Situationen nötig. Sie wird bei der Instanzierung des Prozessschemas auf 0 gesetzt und jeweils beim Start des Knotens um eins erhöht. Das Zustandsübergangsdiagramm der Knotenzustände ist in Abbildung 2.6 zu finden. Die einzelnen Zustände haben jeweils folgende Bedeutung:

**NOT_ACTIVATED** Ausgangszustand eines Knotens. Der Knoten ist momentan nicht ausführbar, da nicht alle seine Startbedingungen erfüllt sind, d. h. nicht alle im Kontrollfluss vorhergehenden Aktivitäten wurden erfolgreich beendet.

**ACTIVATED** Der Knoten steht zur Ausführung bereit. Das heißt, alle Startbedingungen sind erfüllt.

**STARTED** Der Knoten befindet sich momentan in Ausführung.

**SUSPENDED** Der Knoten befindet sich momentan zwar in Ausführung, wurde aber angehalten und kann später fortgesetzt werden.

**COMPLETED** Der Knoten wurde erfolgreich beendet.

**FAILED** Der Knoten konnte nicht erfolgreich beendet werden oder wurde abgebrochen.

**SKIPPED** Der Knoten wird aufgrund des ausgewählten Pfades einer Alternativ-Verzweigung nicht ausgeführt.

Prinzipiell reicht für das Zeitmanagement eine Betrachtung der Meta-Zustände WAITING, RUNNING und TERMINATED aus, jedoch erlaubt die zusätzlich vorgenommene Unterteilung eine genauere Betrachtung des zeitlichen Ablaufs und ein differenzierteres Zeitmanagement. Andererseits sind auch weitere Zustände wie SELECTED denkbar, um auszudrücken, dass ein Benutzer zugesagt hat diesen Schritt zu bearbeiten. Diese weitere Unterteilung ist für das Zeitmanagement nicht notwendig, weswegen der Zustand SELECTED mit dem Zustand ACTIVATED verschmolzen wird (im Unterschied zu [Reichert, 2000]).

Der Ausführungszustand einer Kante besteht lediglich aus dem aktuellen Kantenzustand $\texttt{Es}_e \in \mathbf{ES}$. Hier gibt es die folgenden drei Kantenzustände:

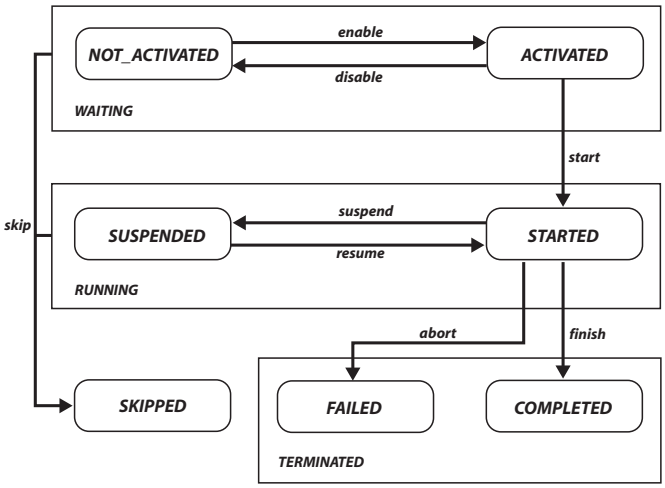



Abbildung 2.6: Zustandsübergangsdiagramm der Knotenzustände (vereinfacht nach [Reichert, 2000])

**NOT_SIGNALLED** Ausgangszustand einer Kante, der Start- und End-Knoten der Kante befindet sich im Zustand NOT_ACTIVATED.

**TRUE_SIGNALLED** Der Startknoten der Kante wurde beendet, das heißt, er befindet sich im Zustand COMPLETED, und die Kante wurde bei einer Alternativ-Verzweigung ausgewählt.

**FALSE_SIGNALLED** Die Kante wird aufgrund der von einer Alternativ-Verzweigung getroffenen Auswahl nicht durchlaufen werden.

Übergänge sind hierbei nur jeweils von NOT_SIGNALLED zu den beiden anderen Zuständen möglich.

Die Ausführungszustände werden bei einer graphischen Darstellung einer Prozessinstanz durch verschiedene Symbole repräsentiert, welche in Abbildung 2.7 zu sehen sind. Dabei werden die Symbole für die Knotenzustände rechts oberhalb des Knotens und die Symbole für die Kantenzustände etwa in der Mitte der Kante platziert.

## 2.2 Zeitmodellierung

Für das Zeitmanagement ist es zunächst nötig, das Basismodell um die Möglichkeit zur Modellierung zeitlicher Eigenschaften zu erweitern. Hierfür stellt die Literatur ([Marjanovic & Orlowska, 1999], [Eder *et al.*, 1999b], [Bettini *et al.*, 2002b], etc.) verschiedene

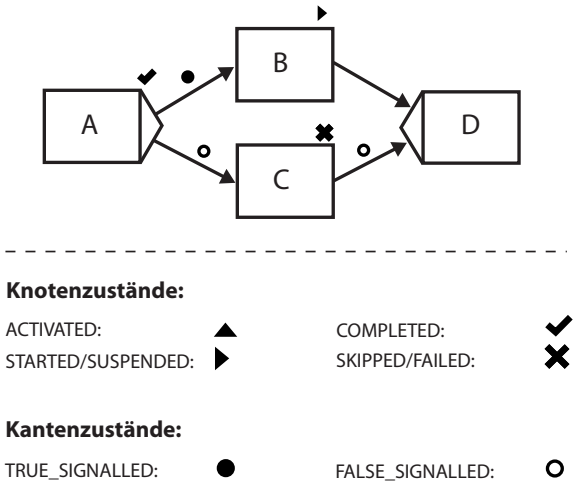



Abbildung 2.7: Graphische Repräsentation der Ausführungszustände

Ansätze bereit. Diese unterscheiden sich, wie schon die Prozess-Metamodelle, in verschiedenen Aspekten wie Ausdrucksmächtigkeit, Komplexität und Semantik. Um die einzelnen Zeitmodelle in Abschnitt 3.1 besser vergleichen zu können, ist es sinnvoll auch hier eine Art Basismodell festzulegen, in dem die zu unterstützenden Zeitkonstrukte definiert und ihre Semantik, soweit ohne Einschränkung möglich, festgelegt wird.

Hierfür müssen zunächst wieder einige für das Verständnis wichtige Begriffe definiert werden (Abschnitt 2.2.1). Im darauf folgenden Abschnitt 2.2.2 werden darauf aufbauend die Grundlagen des Zeitmanagements erörtert, um ein besseres Verständnis der Materie zu erhalten. Ein weiteres Thema ist, in welchen Zeiteinheiten die Zeiteigenschaften eines Prozesses angegeben werden können. Dies wird in Abschnitt 2.2.3 diskutiert.

### 2.2.1 Begriffsbildung

Auch im Bereich der Zeitmodellierung sind einige Begrifflichkeiten zu klären. Dabei geht es vor allem darum, die im Folgenden verwendete Semantik einiger Begriffe festzulegen.

**Zeitpunkt** Ein *Zeitpunkt* ist ein Moment in einem zeitlichen Bezugssystem. Die Lage eines Zeitpunktes wird relativ zum Nullpunkt des Bezugsystems beschrieben, wobei verschiedene Zeiteinheiten (z. B. Minuten, Stunden, Tage) zum Einsatz kommen.

**Kalender** Ein *Kalender* bezeichnet das einer Menge von Terminen zugrunde liegende Bezugssystem. Es handelt sich dabei um eine Festlegung der Jahrrechnung in Jahre sowie deren Unterteilung in Monate, Wochen und Tage.

**Termin** Ein *Termin* ist ein festgelegtes Kalenderdatum inklusive Uhrzeit. Es handelt sich dabei um einen Zeitpunkt, bei dem ein Kalender (z. B. gregorianischer Kalender) als Bezugsystem fungiert.

**Terminierung** Der Begriff *Terminierung* wird in dieser Arbeit, entgegen der normalerweise in der Informatik verwendeten Bedeutung der Endlichkeit der Schritte eines Algorithmus, im Sinne der Festlegung eines Termins verwendet.

**Neuterminierung** Bezeichnet die Veränderung eines zuvor bereits festgelegten Termins.

**Zeitfenster** Ein *Zeitfenster* ist ein für das Auftreten eines bestimmten Ereignisses zur Verfügung stehendes Zeitintervall.

**Start-/Endzeitfenster** *Start-* und *Endzeitfenster* bezeichnen das für den Start beziehungsweise das Ende einer Aktivität erlaubte Zeitfenster.

**Ausführungsfenster** Das *Ausführungsfenster* bezeichnet das für die Ausführung einer Aktivität zur Verfügung stehende Zeitintervall.

**Constraint** Ein *Constraint*[3] schränkt den Wertebereich einer Variablen ein. Dies kann dabei absolut oder relativ zum Wert anderer Variablen geschehen. Constraints treten häufig im Zusammenhang mit sogenannten *Constraint Satisfaction Problemen*[4] [Tsang, 1993] auf. Hier hat man eine Menge von Variablen sowie eine Menge von Constraints auf diesen Variablen und versucht eine Belegung der Variablen derart zu finden, dass sämtliche Constraints erfüllt sind.

**Schedule** Ein *Schedule* ist ein Ablaufplan, ähnlich einem Terminkalender, der verschiedenen Aktivitäten je ein Zeitfenster für ihre Ausführung zuweist.

**Zeitfehler** Ein *Zeitfehler* bezeichnet im Folgenden die Nicht-Einhaltung einer festgelegten Zeitbedingung, beispielsweise aufgrund von Verzögerungen im Betriebsablauf.

**Eskalation** Tritt bei der Ausführung eines Prozesses ein Zeitfehler auf, so kann dieser nicht normal fortgesetzt werden. Es müssen spezielle Schritte zur Korrektur des Problems in die Wege geleitet werden. Hierbei spricht man von *Eskalation*.

### 2.2.2 Zeitkonstrukte

Zeit spielt bei Prozessen an verschiedenen Stellen eine Rolle. Zunächst einmal benötigen die Aktivitäten eines Prozesses eine gewisse Zeit für ihre Bearbeitung, welche natürlich

[3] engl.: Einschränkung
[4] engl.: Bedingungserfüllungsproblem

eingeplant werden muss. Daneben muss es möglich sein, die Abstände zwischen verschiedenen Aktivitäten zu begrenzen, um beispielsweise die Einhaltung von gesetzlichen Mindest-Abständen garantieren zu können. Nicht zuletzt spielen auch Termine für die Aktivitäten eines Prozesses eine Rolle.

In den nächsten 3 Abschnitten werden verschiedene Zeitkonstrukte vorgestellt (siehe Abbildung 2.8), welche die Modellierung der eben erwähnten und weiterer zeitlicher Eigenschaften eines Prozesses erlauben. Dabei wird auch auf die unterschiedlichen Bedeutungen und Semantiken dieser Zeiteigenschaften näher eingegangen, um ein besseres Bild über das Thema Zeit in Prozess-Management-Systemen zu erhalten.

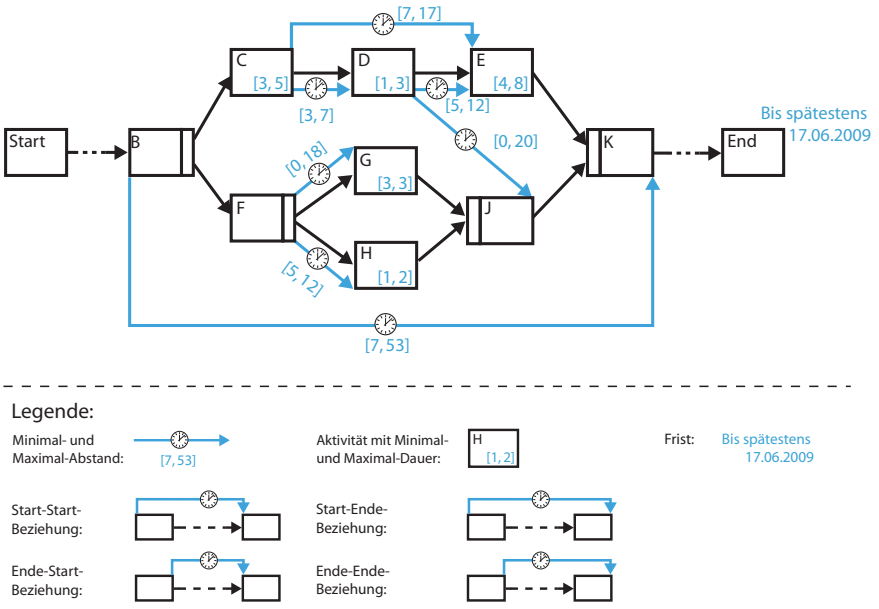



Abbildung 2.8: Zeitkonstrukte

### 2.2.2.1 Dauern

Eines der wichtigsten Elemente bei der Zeitmodellierung von Prozesses ist die Festlegung der *Dauern* der einzelnen Aktivitäten. Hier stellt sich zunächst die Frage, wie man diese ermittelt. In der Regel werden zu diesem Zweck Erfahrungswerte aus früheren Ausführungen der gleichen oder ähnlicher Aktivitäten[5] sowie Schätzungen von Domänenexperten herangezogen. Teilweise ist die erlaubte Dauer aber auch durch externe Vorgaben, wie beispielsweise Gesetze oder Richtlinien festgelegt. Ein Problem, das alle diese Quellen

[5] Diese können beispielsweise aus der Historie des Prozess-Management-Systems extrahiert werden [van der Aalst & Weijters, 2005].

gemein haben, besteht darin, dass selten genaue bzw. eindeutige Daten über die Dauer vorliegen. Zumeist schwankt die Dauer einer Aktivität in gewissen, mehr oder weniger engen Grenzen. Hier stellt sich nun die Frage, wie man die Dauer einer Aktivität im Modell repräsentiert. An dieser Stelle gibt es verschiedene Möglichkeiten, die alle ihre Vor- und Nachteile haben.

Zunächst einmal besteht die Option, bei jeder Aktivität deren *minimale Dauer* $D^{min}$ zu hinterlegen. Dies ist ein sehr optimistischer Ansatz, der vor allem dann sinnvoll ist, wenn Abweichungen von zeitlichen Untergrenzen nur sehr selten vorkommen oder vom System sehr gut toleriert werden können. Ein Nachteil ist, dass auf Basis der minimalen Dauer nur eine sehr ungenaue Zeitkalkulation möglich ist, da die tatsächlichen Dauern der Aktivitäten stark von ihrer minimalen Dauer abweichen. Zusätzlich kann das Prozess-Management-System anhand der minimalen Dauer nur sehr schwer entscheiden, zu welchem Zeitpunkt ein Zeitfehler aufgrund einer Zeitüberschreitung vorliegt.

Ein gegenläufiger Ansatz ist das Hinterlegen der *maximalen Dauer* $D^{max}$ bei jeder Aktivität. Dies wiederum ist ein sehr pessimistischer Ansatz und hat den Vorteil, dass man fast nie mit einer Überschreitung der eingeplanten Zeitspanne rechnen muss. Und selbst dann ist die Wahrscheinlichkeit sehr hoch, dass dieser Engpass durch die nachfolgenden Aktivitäten ausgeglichen und ein Zeitfehler somit vermieden werden kann. Dieses Vorgehen ist vor allem dann empfehlenswert, wenn der Start einer Aktivität vor dem geplanten Termin kaum Schwierigkeiten bereitet. Das heißt, das Prozess-Management-System kann Abweichungen von der zeitlichen Obergrenze sehr gut tolerieren oder die Aktivitäten benötigen fast immer ihre maximale Zeitdauer. Jedoch ist auch hier häufig eine genaue Zeitkalkulation unmöglich, da die tatsächlichen Dauern der Aktivitäten stark von ihrer maximalen Dauer abweichen. Dies spielt jedoch in manchen Situationen keine Rolle und ist in anderen, beispielsweise der Vorbereitung auf ein Ereignis mit festem Termin, möglicherweise sogar von Vorteil.

Eine dritte Möglichkeit ist die Verwendung des *Mittelwertes* $\overline{D}$ der Dauer. Hierbei sind verschiedene Mittelwerte wie arithmetisches Mittel, Median oder Modus denkbar. Dieses Vorgehen hat den Vorteil, dass die Zeitkalkulation relativ genau ist. Dafür kommt es sehr häufig sowohl zu Über- als auch zu Unterschreitungen der eingeplanten Zeitspanne. Somit ist diese Strategie dann sinnvoll einzusetzen, wenn eine genaue Zeitkalkulation wichtig ist, um beispielsweise Vorhersagen treffen zu können und das System gut mit Abweichungen von der kalkulierten Zeit umgehen kann.

Der Einsatz von *Intervallen* $[D^{min}, D^{max}]$ zur Modellierung der Dauer einer Aktivität stellt eine Möglichkeit dar, deren Variabilität Rechnung zu tragen. Über mehrere Instanzen ist die Dauer einer Aktivität letztlich nur sehr selten immer gleich. Daher ist ein Intervall für die Modellierung der Dauer realistischer als die Angabe eines einzelnen Wertes. Ein weiterer Vorteil von Intervallen besteht darin, dass ihre Spanne[6] auch Aufschluss über die zeitlichen Schwankungen der Dauer einer Aktivität liefern kann, was in gewissen

[6] Gemeint ist der Abstand zwischen $D^{min}$ und $D^{max}$.

Planungssituationen von Vorteil ist. Zumeist wird man die minimale und maximale Dauer der Aktivität als Unter- und Obergrenze für das Intervall wählen. Dies ist aber nicht zwingend notwendig und in manchen Fällen kann es sogar von Vorteil sein, die Grenzen des Intervalls abweichend von den Extremwerten zu wählen. Letzteres ist sinnvoll, wenn diese zum Beispiel sehr selten vorkommen oder gar vermieden werden sollen, um Fristen besser einhalten zu können. Dies erhöht jedoch auch die Wahrscheinlichkeit einer Zeitüber- oder Zeitunterschreitung. Die größere Flexibilität und die gewonnenen Vorteile durch die Verwendung von Intervallen erkauft man sich mit einer komplexeren Zeitkalkulation und dem Umstand, dass man als Ergebnis der Berechnung keine festen Zeitpunkte mehr erhält. Stattdessen arbeitet man mit Zeitintervallen, die um so größer werden können, je mehr Aktivitäten, Alternativ-Verzweigungen und Schleifen im Prozessschema enthalten sind (siehe Abbildung 2.9). Letzteres ist jedoch kein wirklicher Nachteil, da die tatsächlichen Werte genauso große Schwankungen aufweisen können, wie die berechneten.

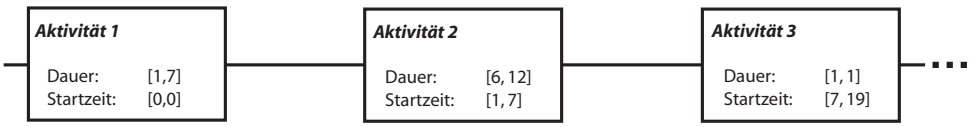



Abbildung 2.9: Ungewissheit über Zeitpunkte durch die Verwendung von Intervallen

Ein weiterer Nachteil bei der Verwendung von Intervallen besteht darin, dass keine Informationen über die Verteilungshäufigkeit innerhalb des gegebenen Intervalls vorliegen. Diesen Umstand haben zum Beispiel Eder und Pichler erkannt und deshalb die sogenannten *Zeit-Histogramme* [Eder & Pichler, 2002] entwickelt. Diese basieren auf einer Intervall-ähnlichen Grundstruktur, wobei Abschnitten des Intervalls unterschiedliche Wahrscheinlichkeiten zugeordnet werden (siehe Abbildung 2.10). Durch diese Erweiterung lässt sich eine noch genauere Zeitkalkulation durchführen, die es sogar erlaubt die Wahrscheinlichkeit dafür zu berechnen, dass eine Aktivität in einem gewissen Zeitintervall ausgeführt wird. Jedoch erkauft man sich diese Mächtigkeit durch einen stark erhöhten Rechenaufwand bei der Zeitkalkulation. Daneben stellt sich die Frage wie die Verteilungshäufigkeiten ermittelt werden können. Die Abschätzungen von Domänenexperten, die bisher hauptsächlich herangezogen wurden, sind hierfür meist zu unpräzise. Es bleibt daher nur die Auswertung der Historie, die jedoch bei der Modellierung eines neuen Prozessschemas noch nicht existiert. Als Mittelweg könnte man mit einer unpräzisen Abschätzung von einem Domänen-Experten beginnen, die dann im Laufe der Zeit mittels Daten aus der gewonnenen Historie verfeinert wird.

Welche der vorgestellten Möglichkeiten zur Anwendung kommt, hängt auch vom eingesetzten Zeit-Algorithmus ab, denn nicht alle Algorithmen sind in der Lage die beiden komplexeren Varianten zur Angabe von Dauern, Intervall und Zeit-Histogramm

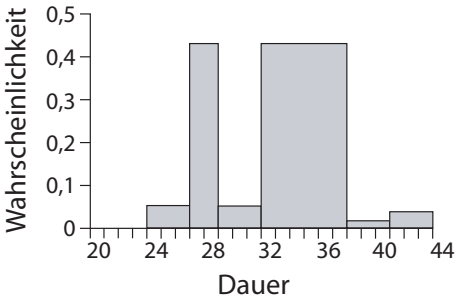



Abbildung 2.10: Zeit-Histogramm

zu verwenden. Ein Vergleich der verschiedenen Arten zur Modellierung der Dauer einer Aktivität ist in Tabelle 2.1 zu finden.

| | Genauigkeit der Zeitkalkulation | Eskalationsrisiko | Komplexität | Besonderheiten |
|---|---|---|---|---|
| minimale Dauer | niedrig | sehr häufig | gering | |
| maximale Dauer | niedrig | selten | gering | |
| mittlere Dauer | mittel - hoch | häufig | gering | Relative genaue Aussage über Zeitpunkt |
| Intervalle | mittel | selten | mittel | Grenzen können angepasst werden um zeitliche Ziele zu forcieren |
| Zeit-Histogramme | sehr hoch | selten | hoch | Wahrscheinlichkeitsaussage über den Zeitpunkt von Ereignissen möglich |

Tabelle 2.1: Vergleich der Modellierungsarten von Aktivitätsdauern

Wie erwähnt, soll die zu entwickelnde Zeitmanagementkomponente Unterstützung für einen Eskalations-Mechanismus bieten. Da eine Eskalation immer eine Ausnahmesituation in der Ausführung des Prozesses darstellt, sollte das Auftreten der Eskalation auslösenden Zeitfehler auf ein nötiges Minimum reduziert werden. Weiter muss, um ein ausgereiftes Zeitmanagement bieten zu können, sowohl bei Über- als auch bei Unterschreitung der eingeplanten Dauer einer Aktivität eine Eskalation ausgelöst werden. Daher sind Intervalle

für uns die beste Wahl zur Modellierung der Dauer einer Aktivität. Zum einen lässt sich mit diesen das Eskalationsrisiko auf ein nötiges Minimum reduzieren und zum anderen lassen sich Berechnungen mit Intervallen noch mit vernünftigem Rechenaufwand durchführen. Dies ist wichtig, um auch zur Laufzeit der Prozesse eine Unterstützung durch das Zeitmanagement bieten zu können. Für die Vorhersage von Startzeitpunkten oder Ähnlichem sind Intervalle jedoch von Nachteil, da sich mit diesen unter Umständen leider nur ein sehr grober zeitlicher Rahmen abstecken lässt. Es kann daher sinnvoll sein, die Intervalle noch zusätzlich mit einem Mittelwert zu kombinieren, um möglichst präzise Vorhersagen treffen zu können.

Ein bisher nicht berücksichtigter Punkt ist das Problem, dass die Dauer einer Aktivität zum Modellierungszeitpunkt eventuell nicht bekannt ist, da sie von exogenen Einflüssen, beispielsweise Lieferzeiten, abhängt oder durch verschiedene Entscheidungen im vorhergehenden Teil des Prozesses, beispielsweise an XOR-Blöcken, stark beeinflusst wird. Dieses Problem lässt sich nicht ohne Weiteres lösen. Wenn bereits zum Modellierungszeitpunkt eine Überprüfung der Zeitbedingungen stattfinden soll, besteht die einzige Möglichkeit darin, die Grenzen des Intervalls großzügig genug zu wählen und diese dann zur *Laufzeit*, sobald die tatsächliche Dauer bekannt ist, zu adaptieren. Dies funktioniert meist ohne Weiteres, da Einschränkungen eines Intervalls für die meisten Algorithmen zum Zeitmanagement kein Problem darstellen. Eine Ausweitung könnte dagegen eventuell problematische Folgen haben. Eine ausführliche Diskussion dieser Thematik findet an geeigneter Stelle in Abschnitt 6.2.2 statt.

#### 2.2.2.2 Zeitliche Beziehungen

Die Angabe der Dauern der Aktivitäten reicht für das Zeitmanagement nicht aus. So muss es zusätzlich möglich sein, Einschränkungen für die Abstände zwischen Aktivitäten zu modellieren. Hier sind zunächst die vier möglichen *Abstandsbeziehungen* zu beachten. Die am häufigsten vorkommende *Ende-Start*-Beziehung legt den zeitlichen Abstand zwischen dem Ende der einen und dem Start der anderen Aktivität fest. Die seltener vorkommende *Start-Start*-Beziehung legt den Zeitabstand zwischen dem Start der beiden Aktivitäten fest. Analoges gilt für die *Start-Ende*- und die *Ende-Ende*-Beziehung. Dabei haben alle Beziehungen, außer der Ende-Start-Beziehung, die indirekte Eigenschaft auch die Dauer einer oder beider der beteiligten Aktivitäten zu begrenzen. Die vier Abstandsbeziehungen sind zur besseren Verdeutlichung nochmals in Abbildung 2.11 dargestellt.

##### Abstände

Mit diesen vier Abstandsbeziehungen lassen sich die Abstände zwischen verschiedenen Ereignissen, das heißt dem Start oder Ende von Aktivitäten, einschränken. Dabei muss es sowohl möglich sein den Abstand nach oben, als auch nach unten zu begrenzen. Hierbei spricht man von Minimal- und Maximal-Abstand. Ein *Minimal-Abstand* $Z_{ij}^{min}$ zwischen zwei Aktivitäten $A_i$ und $A_j$ legt fest, dass zwischen dem Start oder Ende der

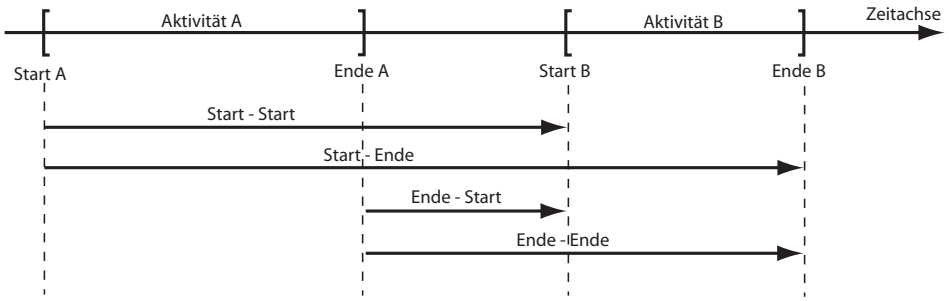



Abbildung 2.11: Die vier Abstandsbeziehungen

Aktivität $A_i$ und dem Start oder Ende der Aktivität $A_j$ eine gewisse Zeitspanne vergehen muss. Dies ist beispielsweise nötig, um zu modellieren, dass zwischen dem Verabreichen eines Narkotikums und dem Einsetzen der Wirkung, das heißt dem möglichen Beginn der Operation, eine gewisse Zeit vergeht. Ein *Maximal-Abstand* $Z_{ij}^{max}$ zwischen zwei Aktivitäten $A_i$ und $A_j$ hingegen legt fest, dass zwischen Start oder Ende von $A_i$ und Start oder Ende von $A_j$ höchsten eine gewisse Zeit vergehen darf. So lässt beispielsweise die Wirkung des Narkotikums nach einer gewissen Zeit wieder nach, weswegen die Operation bis zu diesem Zeitpunkt beendet sein muss. An diesem Beispiel sieht man auch, wie sinnvoll die im vorherigen Abschnitt eingeführten Abstandsbeziehungen sind. Bei dem Beispiel für den Minimal-Abstand handelt es sich um eine Ende-Start-Beziehung und bei dem für den Maximal-Abstand um eine Ende-Ende-Beziehung.

Eine kompakte Methode zur Repräsentation von Minimal- und Maximal-Abstand ist die Verwendung von *Intervallen* $[Z_{ij}^{min}, Z_{ij}^{max}]$. Diese empfiehlt sich vor allem im Zusammenhang mit der Verwendung von Intervallen zur Repräsentation von Dauern. Damit erreicht man sowohl eine einheitliche Darstellung der Zeit, als auch eine einheitliche Zeitkalkulation. Bei *Intervall-Abständen* bedeutet eine Intervall-Untergrenze von $-\infty$ das kein Minimal-Abstand gegeben ist und eine Intervall-Obergrenze von $\infty$ das kein Maximal-Abstand gegeben ist. Ein weiterer Punkt für die Verwendung von Intervallen zur Repräsentation des Abstandes ergibt sich auch aus dem bereits in Abschnitt 2.1.2.3 skizzierten Umstand, dass für eine korrekte und vollständige Zeitkalkulation zumindest ein impliziter Minimal- und Maximal-Abstand zwischen zwei im Kontrollfluss aufeinanderfolgenden Aktivitäten nötig ist. Führt man diesen Gedanken weiter, so bedeutet dies, dass zwischen zwei Aktivitäten zumeist sowohl Minimal- als auch Maximal-Abstand spezifiziert werden sollten, was für eine kompakte Repräsentation beider durch ein Intervall spricht.

Auch bei der Angabe von Abständen besteht das Problem, dass die konkreten Werte manchmal erst zur Laufzeit feststehen. Doch ergeben sich hieraus keine weiteren Schwierigkeiten, da, wie bei den Dauern erwähnt, eine zusätzliche Einschränkung für die

Zeitkalkulation kein Problem darstellt. Daher wird man an einer solchen Stelle zunächst ein Intervall $[-\infty, \infty]$ verwenden, dass dann zur Laufzeit weiter eingeschränkt wird, sobald die konkreten Werte feststehen.

#### 2.2.2.3 Fristen

Das letzte hier zur Modellierung der Zeitbedingungen eines Prozesses benutzte Element sind *Fristen*. Diese werden oft, wenn auch nicht ganz korrekt, als *Deadline* bezeichnet. Eine Frist $D_i$ legt den Zeitpunkt für das Eintreten eines Ereignisses $i$, also Start oder Ende einer Aktivität, fest. Dabei kann es sich sowohl um den frühesten möglichen Zeitpunkt handeln, ab dem das Ereignis eintreten darf, als auch um den spätesten möglichen Zeitpunkt, bis zu dem Es eingetreten sein muss. Bei einer Deadline hingegen handelt es sich korrekterweise lediglich um den spätesten möglichen Zeitpunkt für das Ende einer Aktivität. Beispiele für Fristen sind ein Arzt-Termin, hierbei handelt es sich, zumindest aus Sicht des Patienten, um einen frühestmöglichen Start-Zeitpunkt, oder der Abgabetermin einer Diplomarbeit, wobei es sich meist um den spätestmöglichen Zeitpunkt handelt.

Fristen sind ein wichtiges Thema in der Zeitmodellierung. Dabei besteht das Problem, dass Fristen meist erst zur Laufzeit des Prozesses feststehen. Zur Modellierungszeit ist ihre Angabe, wenn überhaupt, meist nur in abstrakter Form möglich, etwa „jeder erste Tag des Monats“. Zugleich ist es aber von zentraler Bedeutung, dass so früh wie möglich, demnach am besten zur Modellierungszeit, überprüft wird, ob die Fristen überhaupt eingehalten werden können. Außerdem ergeben sich aus Fristen häufig Einschränkungen für die Startzeitpunkte anderer Aktivitäten. So kann beispielsweise aus dem Abgabetermin einer Diplomarbeit darauf geschlossen werden, wann diese spätestens in Druck gegeben werden sollte.

Im Zusammenhang mit Fristen wird häufig von *absoluten Zeitdaten* gesprochen, im Gegensatz dazu werden Dauern und Abstände zwischen Aktivitäten als *relative Zeitdaten* bezeichnet. Diese Unterscheidung ist jedoch künstlich, da Fristen immer in Abstände zu einem virtuellen „Nullzeitpunkt“ umgewandelt werden können. Eine Frist wird damit zu einem normalen Abstand zwischen diesem „Nullzeitpunkt“, häufig durch eine Start-Aktivität dargestellt, und der Aktivität, die durch die Frist betroffen ist. Der zuvor als Beispiel verwendete Arzttermin wird damit zu einem Ende-Start Minimal-Abstand zwischen der Start-Aktivität und der, den Arztbesuch repräsentierenden, Aktivität und der Abgabetermin wird zu einem Ende-Ende Maximal-Abstand ebenfalls zwischen der Start-Aktivität und der, die Abgabe repräsentierenden, Aktivität.

### 2.2.3 Zeiteinheiten

Bei den bisherigen Betrachtungen wurde noch nicht beachtet, in welcher *Zeiteinheit* / welchen *Zeiteinheiten* Dauern und Abstände angegeben werden. Manche Aktivitäten dauern nur wenige Minuten, andere mehrere Wochen oder gar Monate. Es ist leicht

einzusehen, dass es nicht unbedingt sinnvoll ist, für alle Zeitangaben die gleiche Zeiteinheit zu verwenden. Einerseits ist es nur schwer möglich eine Dauer von wenigen Minuten in Wochen oder gar Monaten auszudrücken, andererseits führt es auch zu Problemen, wenn Monate in Minuten ausgedrückt werden sollen, schon allein deswegen, weil nicht alle Monate die gleiche Länge haben. Doch selbst wenn dies ignoriert werden könnte, ist die Angabe eines Maximal-Abstandes von einem Monat in Minuten nur wenig sinnvoll. In diesem Fall führt eine Überschreitung um eine Minute bereits zu einem Zeitfehler, obwohl sie eigentlich keine Rolle spielt. Ein weiteres Problem ist, dass etwa ein Maximal-Abstand von 0 Tagen (d. h. beide Ereignisse müssen am selben Kalendertag stattfinden) nicht in Stunden oder Minuten ausgedrückt werden kann. Am ehesten entspräche dies einem Maximal-Abstand von 1439 Minuten, jedoch ist damit nicht garantiert, dass beide Ereignisse am selben Tag stattfinden. Daher sollte das Prozess-Management-System in der Lage sein, mit Zeitangaben verschiedener *Granularitäten* wie Sekunden, Minuten, Stunden etc. umzugehen.

## 2.3 Fazit

Wie diskutiert, gibt es für das Zeitmanagement einige nicht-triviale Anforderungen. Dabei stellen vor allem die beiden Konstrukte Alternativ-Verzweigungen und Schleifen große Probleme dar, welche nicht einfach zu lösen sind. Aber auch die Anforderungen der Zeitmodellierung selbst lassen einige Fragestellungen offen, die es zu klären gilt. So zum Beispiel, wie Fristen behandelt werden oder inwiefern Zeitbedingungen, das heißt, Fristen, Dauern und Abstände, die erst zur Laufzeit feststehen, unterstützt werden können.

Zunächst soll daher im folgenden Kapitel die einschlägige Literatur auf Möglichkeiten zur Zeitmodellierung und Lösungen für die erwähnten Probleme hin untersucht werden. Dabei wird das Augenmerk nicht nur auf einer Sichtung des aktuellen Stands der Technik, sondern vielmehr auf einem kritischen Vergleich der untersuchten Möglichkeiten liegen. In den darauf folgenden Kapiteln wird dann ein eigener Ansatz zur Integration der geforderten Zeit-Konstrukte in ein Prozess-Metamodell vorgestellt, sowie einige Lösungsansätze für die erwähnten Probleme umfassend diskutiert.

# 3 Zeit und Prozess-Management-Systeme – Stand der Technik

Zeit spielt in vielen Bereichen der Informatik eine Rolle. Insbesondere die Bereiche Künstliche Intelligenz (KI), Datenbank- und Informationssysteme sowie Rechnernetze beschäftigen sich schon seit Längerem mit dem Thema Zeit. Auch in anderen Wissensgebieten, wie Operations Research (Mathematik), Projektmanagement, Physik oder Wirtschaftswissenschaften, hat man sich bereits mit diesem Thema beschäftigt. Dabei stellt sich meist nicht die Frage „Was ist Zeit“ (außer vielleicht in der Physik), sondern wie kann man zeitabhängige Vorgänge formal beschreiben, analysieren und daraus Schlussfolgerungen ableiten.

In diesem Kapitel werden einige relevante, zeitbezogene Forschungsarbeiten vorgestellt. Die meisten davon stammen, schon aufgrund der Natur dieser Arbeit, aus dem Bereich der Informationssysteme. Aber auch andere Bereiche wie beispielsweise Künstliche Intelligenz oder Projektmanagement sind vertreten.

Dieses Kapitel gliedert sich wie folgt: In Abschnitt 3.1 werden zunächst unterschiedliche Möglichkeiten zur Darstellung und Analyse temporalen Wissens diskutiert. Vorgestellt wird dabei eine Auswahl von für das Prozess-Management relevanten Arbeiten.

Abschnitt 3.2 beschreibt verschiedene weiterführende Konzepte. Teilweise stehen diese Themen nur am Rand mit dem Zeitmanagement in Verbindung, andere hingegen beschäftigen sich mit zentralen Bereichen des Zeitmanagements.

Daran anschließend (Abschnitt 3.3) werden einige an das Zeitmanagement angrenzende Themenbereiche betrachtet. Hierbei handelt es sich hauptsächlich um Themen wie Scheduling oder Ressourcenmanagement, die von einem funktionierenden Zeitmanagement abhängig sind.

Abgeschlossen wird dieses Kapitel durch eine kurze Zusammenfassung (Abschnitt 3.4), welche nochmals einen Überblick über die wichtigsten Themen gibt.

## 3.1 Repräsentation von Zeit

Die Repräsentation von Zeit im Zusammenhang mit Prozess-Management-Systemen wird schon seit Anfang der 90er Jahre von verschiedenen Gruppen erforscht. Im Laufe der Jahre wurden dabei diverse Formalismen zur Repräsentation des zu einem Prozess gehörenden temporalen Wissens vorgeschlagen. Diese lassen sich verschiedenen Paradigmen mit unterschiedlichen Ursprüngen und Anforderungen zuordnen.

Zunächst werden verschiedene Formalismen betrachtet, die zur Gruppe der *Netzplantechniken* [Rinza, 1994] gehören (Abschnitt 3.1.1). Dabei handelt es sich um einen graphbasierten Ansatz zur Repräsentation zeitlicher Abhängigkeiten und Bedingungen, der vor allem für die Planung und Steuerung von Projekten entwickelt wurde. Einige der Methoden, die zur Gruppe der Netzplantechniken zählen, wurden bereits Mitte der 50er Jahren entwickelt und beispielsweise schon früh bei der US-Marine zur Planung von Fertigungsprozessen verwendet [Sapolsky, 1972].

Das zweite vorgestellte Paradigma ist die Gruppe der sogenannten *Temporal Constraint Networks* [Dechter *et al.*, 1991] (Abschnitt 3.1.2). Aufgabe dieser, hauptsächlich in der Künstlichen Intelligenz und dem Operations Research anzutreffenden, Problemstellungen ist es, eine Belegung für Variablen zu finden. Dabei müssen eine Reihe von Bedingungen erfüllt werden, welche an die Werte der Variablen geknüpft sind.

Die dritte Gruppe basiert auf Petri-Netzen [Petri, 1962] (Abschnitt 3.1.3), einem mathematischen Modell zur Modellierung nebenläufiger Systeme. Auch einige Prozess-Metamodelle verwenden Petri-Netze als Grundlagen zur Modellierung der Prozesse, woraus sich Synergieeffekte ergeben.

Im vorletzten Abschnitt 3.1.4 werden verschiedene weitere Formalismen vorgestellt, welche sich keinem der zuvor vorgestellten Paradigmen zuordnen lassen.

Abschließend folgt ein kritischer Vergleich der vorgestellten Verfahren (Abschnitt 3.1.5), bei dem insbesondere auf die Tauglichkeit der verschiedenen Paradigmen und Formalismen für die Repräsentation der Zeiteigenschaften von Prozessen eingegangen wird.

### 3.1.1 Netzplantechnik

Der Begriff der *Netzplantechnik* beinhaltet „alle Verfahren zur Analyse, Beschreibung, Planung, Steuerung und Überwachung von Abläufen auf der Grundlage der Graphentheorie, wobei Zeit, Kosten, Einsatzmittel bzw. Ressourcen berücksichtigt werden können“ [DIN, 1987]. Ziel der Netzplantechnik in unserem Anwendungsgebiet ist die Planung der logischen Beziehungen zwischen Aktivitäten und deren zeitliche Lage.

Basis der Netzplantechnik ist ein sogenannter *Netzplan*. Dies ist ein Graph welcher die unterschiedlichen Vorgänge sowie deren logische Beziehungen repräsentiert. Darauf aufbauend werden die unterschiedlichen Fragestellungen der betrachteten Planungsprobleme auf spezielle graphentheoretische Fragestellungen zurückgeführt (beispielsweise die Suche nach dem kürzesten Pfad). Der Aufbau des Netzplanes variiert dabei von Verfahren zu Verfahren (siehe Abbildungen 3.2, 3.3 und 3.4) und spiegelt damit die unterschiedlichen Betrachtungsweisen der Techniken wieder.

Das Grundprinzip der Netzplantechnik beruht auf der Erkenntnis, dass wenige Aktivitäten den Verlauf des gesamten Projektes bestimmen [Saynisch *et al.*, 1979], [Rinza, 1994]. Diese Aktivitäten werden auch als kritische Aktivitäten bezeichnet. Sie bilden zusammen

den längsten Pfad durch den Netzplan[1], welcher auch als *kritischer Pfad* bezeichnet wird (siehe Abbildung 3.1). Die Aufgabe der Algorithmen ist es nun diesen kritischen Pfad zu bestimmen, da dieser die Gesamtzeit des Projektes maßgeblich beeinflusst. Unkritische Aktivitäten hingegen können umgeplant werden, ohne dass der Gesamtplan verändert werden muss.

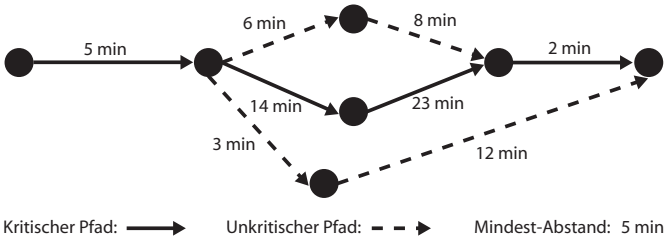



Abbildung 3.1: Kritischer Pfad durch einen Netzplan

Eine der ältesten und bekanntesten Netzplantechniken ist die *Critical Path Method* (*CPM*), welche in Abschnitt 3.1.1.1 vorgestellt wird. Zur etwa gleichen Zeit wie CPM wurde die *Program Evaluation and Review Technique* (*PERT*) entwickelt, welche auch technisch sehr eng mit CPM verwandt ist. Diese wird in Abschnitt 3.1.1.2 vorgestellt. Abschnitt 3.1.1.3 behandelt die *Metra Potential Method* (*MPM*), eine weitere Methode der Netzplantechnik.

Einige andere hier nicht diskutierte Verfahren der Netzplantechnik sind beispielsweise die *Graphical Evaluation and Review Technique* (GERT) oder die *Generalized Activity Networks* (GAN). Diese Verfahren und weitergehende Fragestellungen zur Netzplantechnik, welche über das hier Besprochene hinaus gehen, werden beispielsweise in [Rinza, 1994] oder [Neumann & Morlock, 1993] diskutiert.

#### 3.1.1.1 CPM (Critical Path Method)

Die *Critical Path Method* basiert auf einem *Vorgangs-Pfeil-Netz* [Shaffer *et al.*, 1965]. Im CPM-Netzplan werden Aktivitäten und Anordnungsbeziehungen als Pfeile und die Ereignisse als Knoten dargestellt. Die Pfeile werden mit den Dauern der Aktivitäten beziehungsweise der Anordnungsbeziehungen beschriftet. Abbildung 3.2 zeigt ein solches Vorgangs-Pfeil-Netz.

Durch CPM-Graphen kann explizit eine Ende-Start-Abstandsbeziehung modelliert werden. Implizit ist es, beispielsweise durch Hinzufügen eines AND-Split, auch möglich eine Start-Start-Abstandsbeziehungen zu modellieren. Eine Modellierung der anderen beiden Abstandsbeziehungstypen ist nicht möglich. Darüber hinaus sind zur Modellierung der zeitlichen Abhängigkeiten zweier nicht sequenziell ablaufender Aktivitäten zusätzlich

[1] Dieser ist nicht notwendigerweise eindeutig.

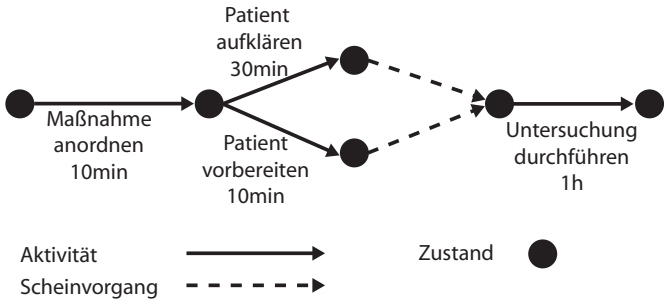



Abbildung 3.2: Vorgangs-Pfeil-Netz

sogenannte Scheinvorgänge, das heißt fiktive Vorgänge der Dauer 0, nötig (siehe Abbildung 3.2).

Zur Berechnung der Zeiten eines CPM-Graphen können ein modifizierter Bellmann- [Bellmann, 1958] oder ein Shortest-Path-Partitioning-Algorithmus [Evans & Edward, 1992] verwendet werden, welche beide die Laufzeitkomplexität $O(n^2)$ besitzen. Diese berechnen die frühesten Startzeitpunkte der beteiligten Aktivitäten sowie die Laufzeit des kritischen Pfades und den kritischen Pfad selbst.

Mit CPM-Graphen ist es nicht möglich, Alternativ-Verzweigungen und Schleifen zu modellieren. Außerdem können keine Maximal-Abstände und -Dauern repräsentiert werden. Damit ist auch eine Repräsentation von Fristen ausgeschlossen. Dies führt gleichzeitig dazu, das CPM-Graphen niemals inkonsistent sein können.

#### Das Modell von Marjanovic und Orlowska

Ein auf der Critical Path Method basierender Ansatz für das Prozess-Management wird in [Marjanovic & Orlowska, 1999] entwickelt. Dauern werden dabei, entgegen der ursprünglichen CPM, als Intervalle repräsentiert. Hierzu wird der Shortest-Path-Partitioning-Algorithmus erweitert, um auf Basis der Unter- und Obergrenze der Intervalle die minimale und maximale Dauer des Prozesses zu berechnen. Die Berechnung der Dauer erfolgt dabei lediglich anhand der Zeitdaten des Kontrollflusses, eventuell vorhandene zusätzliche Zeitbedingungen werden zunächst ignoriert. Als Ergebnis dieser Berechnung erhält man, neben der minimalen und maximalen Gesamtlaufzeit des Prozesses, für jeden Knoten einen frühesten und einen spätesten Startzeitpunkt.

Zur Unterstützung von Alternativ-Verzweigungen werden der kürzeste und der längste Instanztyp getrennt betrachtet. Hierzu werden zwei Variationen des Berechnungsalgorithmus verwendet, wovon eine die Werte des kürzesten und die andere die Werte des längsten Instanztypen berechnet.

In einem weiteren Verifikationsschritt wird, getrennt für jeden der beiden Instanztypen, auf Basis der berechneten Werte die Gültigkeit der zuvor nicht beachteten Zeitbedingungen

überprüft. An Zeitbedingungen werden dabei sowohl Minimal-Abstände als auch Maximal-Abstände sowie Fristen unterstützt. Zur Überprüfung der Gültigkeit werden für jede Zeitbedingung diverse Voraussetzungen überprüft, welche sich aus der Art der jeweiligen Zeitbedingung ableiten. Beispielsweise muss für eine Deadline sichergestellt werden, dass der späteste Zeitpunkt der Aktivität im längsten Instanztypen noch vor dem angegeben Datum liegt. Bei diesem Schritt werden auch Abhängigkeiten zwischen verschiedenen Zeitbedingungen (z. B. Abstände zwischen Fristen) auf ihre Erfüllbarkeit hin überprüft. Eventuelle Auswirkungen der Zeitbedingungen auf die berechneten Zeiten werden nicht auf die berechneten Zeitwerte übertragen und daher auch nicht weiter beachtet.

#### Semantische Alternativmodellierung

*SAM* (*Semantische Alternativmodellierung*) [Braig, 2004] ist ein weiterer auf der CPM beruhender Ansatz für das Prozess-Management, für welchen bisher jedoch nur theoretische Überlegungen existieren. Er besitzt einige Gemeinsamkeiten zu dem vorgestellten Ansatz von Marjanovic und Orlowska. Es werden Intervalle für die Repräsentation der Dauern verwendet und darauf aufbauend die minimale und maximale Dauer des Prozesses berechnet. Zur Unterstützung von Alternativ-Verzweigungen werden ebenfalls der kürzeste und der längste Instanztyp getrennt berechnet bzw. betrachtet. Maximal-Dauern und -Abstände werden auf ähnliche Weise wie bei Marjanovic und Orlowska durch eine zusätzliche Verifikation der Zeitbedingungen unterstützt, jedoch ist diese bereits in den Berechnungsalgorithmus integriert. Das heißt, die Berechnung der Zeitdaten auf Basis des Kontrollflusses und die Verifikation der zusätzlichen Zeitbedingungen finden zeitgleich statt.

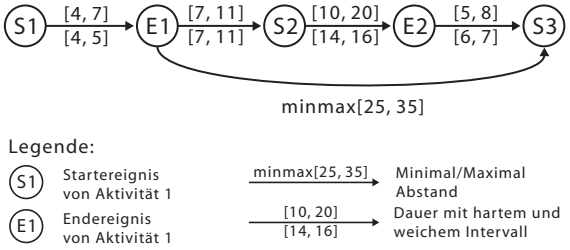
Eine Besonderheit dieses Ansatzes ist, dass für die Dauern und Abstände nicht nur ein, sondern zwei Intervalle angegeben werden, ein sogenanntes hartes und ein weiches Intervall. Dabei muss gelten, dass das weiche Intervall Teilmenge des harten ist. Die Idee ist dabei Folgende: Stellt der Algorithmus bei der Zeitplanung fest, dass das harte Intervall aufgrund von Zeitbedingungen nicht einzuhalten ist, kann er dieses bis auf die Grenzen des weichen Intervalls einschränken (siehe Beispiel 3.1). Die konkreten Einschränkungen werden vom Algorithmus jedoch nicht berechnet, es wird einzig festgestellt, ob die nötigen Einschränkungen im erlaubten Bereich liegen oder nicht. Basierend auf dieser Einschätzung werden dem Zeitmodell des Prozessschemas verschiedene Konsistenz-Stufen (*Solveable*, *UnsafetyWorstCase*, *Unsafety*)[2] zugewiesen.

#### Das Modell von Eder, Panagos und Rabinovich

Ein dritter Ansatz für das Prozess-Management auf der Basis von CPM wird in [Eder *et al.*, 1999a] vorgestellt. Dauern werden dabei als ein fester Wert spezifiziert, wobei nicht angegeben ist, ob es sich dabei um Minimum, Maximum oder einen Mittelwert handelt. Auf Basis dieser Dauern und der vorhandenen Minimal-Abstände werden in zwei Schritten

[2] engl.: Lösbar; Unsicher im schlimmsten Fall; Unsicher

Lediglich auf Basis der angegeben harten Intervalle sind der minimal und maximal Abstand zwischen E1 und S3 nicht einzuhalten ($[7, 11] + [10, 20] + [5, 8] = [22, 39]$).




Jedoch erlauben die angegeben weichen Intervalle eine Einschränkung des Abstandes zwischen E1 und S3 bis auf $[27, 34]$, womit sowohl der minimal als auch der maximal Abstand eingehalten werden kann. Dabei ist es für die Überprüfung der Konsistenz unerheblich, ob lediglich der Abstand zwischen S2 und E2 auf $[13, 16]$ eingeschränkt wird, oder ob beispielsweise der Abstand zwischen S1 und E2 auf $[12, 17]$ und der Abstand zwischen E2 und S3 auf $[6, 7]$ eingeschränkt wird.
Im Extremfall kann es jedoch dazu kommen, dass sowohl Aktivität 2 als auch der Abstand zwischen Aktivität 2 und 3 die maximale Dauer des harten Intervalls benötigen.Daher wird dem Prozessschema in diesem Fall der Konsistenz-Zustand *UnsafetyWorstCase* zugewiesen und nicht *Solveable* wie im Fall eines ohne Einschränkung gültigen Prozessschemas.

Beispiel 3.1: Semantische Alternativmodellierung

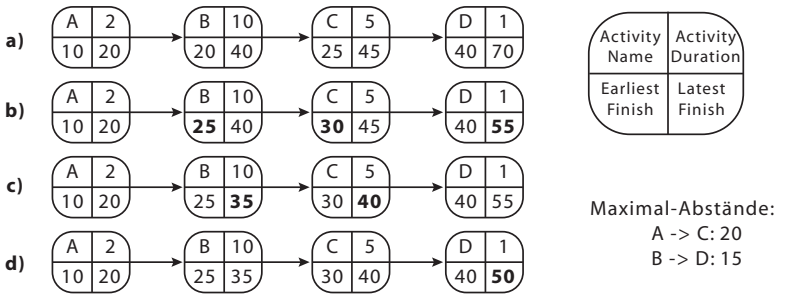
die frühesten und spätesten Endzeitpunkte der Aktivitäten (über alle Instanztypen) berechnet. Dabei werden Fristen für die Berechnungen zum Modellierungszeitpunkt zunächst durch Worst-Case-Abschätzungen in Minimal-Abstände transformiert. Erst zur Laufzeit werden die Fristen dann auch als solche betrachtet, wenn der Startzeitpunkt der Prozessinstanz bekannt ist.

Auf Basis der berechneten Endzeitfenster wird in einem dritten Schritt die Einhaltung der Maximal-Abstände überprüft, wobei versucht wird, die frühesten und spätesten Endzeitpunkte der Aktivitäten entsprechend den Bedingungen anzupassen. Das heißt, die frühsten und spätesten Endzeitpunkte werden entsprechend des Maximal-Abstandes nach vorne oder hinten verschoben (siehe Beispiel 3.2). Danach findet nochmals eine Aktualisierung der Zeitdaten statt, um die durch die Integration der Maximal-Abstände entstandenen Änderungen vollständig einzubeziehen.

#### 3.1.1.2 PERT

Die *Ereignis-Knoten-Netze*, auf denen die *Program Evaluation and Review Technique* basiert, unterscheiden sich nur unwesentlich von den Vorgangs-Pfeil-Netzen der CP-

Folgendes Beispiel aus [Eder *et al.*, 1999a] verdeutlicht das Vorgehen bei der Integration von Maximal-Abständen in die berechneten Zeitwerte.




a) zeigt das ursprüngliche Prozessschema mit den 4 Aktivitäten A, B, C und D sowie den berechneten Endzeitpunkten. Bei der Integration des Maximal-Abstandes $B \rightarrow D$ werden die Endzeitpunkte, wie in b) gezeigt, angepasst. Wird der Maximal-Abstand $A \rightarrow C$ integriert, wird wie in c) zu erkennen der Maximal-Abstand $B \rightarrow D$ in den spätesten Endzeitpunkten verletzt. Durch einen weiteren Anpassungsschritt erhält man d), bei dem alle Maximal-Abstände erfolgreich integriert sind.

Beispiel 3.2: Integration von Maximal-Abständen beim Modell von Eder, Panagos und Rabinovich [Eder *et al.*, 1999a]

Methode (siehe Abbildung 3.3). Der Hauptunterschied liegt darin, dass bei CPM Vorgänge und bei PERT Ereignisse im Mittelpunkt des Interesses stehen. Bei PERT liegt das Augenmerk damit eher auf der Kontrolle des Projektfortschritts und weniger auf der Steuerung der einzelnen Vorgänge [Luttman *et al.*, 1995].

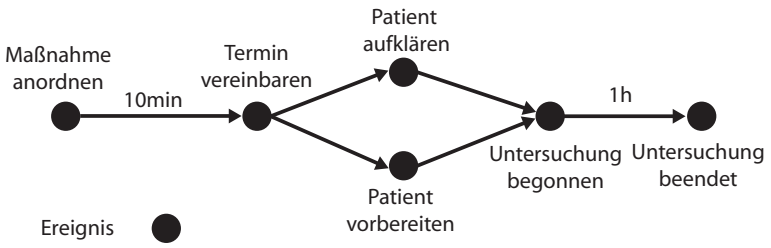



Abbildung 3.3: Ereignis-Knoten-Netz

Die Vorgehensweise der beiden Techniken unterscheidet sich vor allem durch eine stochastische Schätzung der Zeitdaten auf Seiten von PERT. Damit geht auch eine leicht veränderte Berechnung der Zeitfenster der Aktivitäten einher. Bei PERT wird die Dauer

einer Aktivität durch drei Schätzwerte festgelegt: eine optimistische Schätzung *bc*, ein häufigster Wert *ac* und eine pessimistische Schätzung *wc*. Aus diesen wird unter der Annahme einer $\beta$-Verteilung der Mittelwert $\mu = \frac{1}{6}(bc + 4 * ac + wc)$ und die Varianz $\sigma^2 = \frac{1}{36}(wc - bc)^2$ der Dauer berechnet. Die Annahme der $\beta$-Verteilung gilt dabei nur für die Berechnung der Zufallsparameter, für die Ermittlung der Zeitdaten und die spätere Analyse werden diese dann als normalverteilt betrachtet.

Ein weiterer Unterschied zwischen CPM und PERT ist, dass bei PERT Aktivitäten in ein Anfangs- und ein Endereignis aufgeteilt werden können (siehe Aktivität Untersuchung in Abbildung 3.3). Dadurch ist es möglich, alle vier Abstandsbeziehungen in einem PERT-Graphen darzustellen.

Die Berechnung der Zeitdauern kann nun durch dieselben Algorithmen wie bei CPM erfolgen, wobei der Mittelwert die Rolle der Zeitdauern des CPM-Netzes übernimmt. Zusätzlich werden durch dieselben Berechnungsvorschriften die akkumulierten Varianzen berechnet. Als Ergebnis erhält man Erwartungswerte für die Start-Termine der Aktivitäten und die zugehörigen Varianzen. Damit lassen sich Wahrscheinlichkeitsaussagen über die Zeitpunkte und Zeitabstände der Ereignisse treffen. Es ist jedoch nicht möglich, genaue früheste und späteste Startzeitpunkte zu berechnen. Damit ist es auch sehr schwer zu bestimmen, zu welchem Zeitpunkt ein Zeitfehler ausgelöst werden muss.

In PERT können keine Maximal-Abstände und -Dauern repräsentiert werden, ebenso wenig ist es möglich, Alternativ-Verzweigungen und Schleifen zu modellieren.

#### ePERT

Pozewaunig und Eder haben für das Prozess-Management ein auf PERT basierendes Modell namens extended PERT (ePERT) entwickelt [Pozewaunig, 1996], [Pozewaunig *et al.*, 1997], welches in der Lage ist, auch Alternativ-Verzweigungen darzustellen und zu behandeln. Aufbauend auf der berechneten mittleren Dauer $\mu$ der Aktivitäten berechnen sie für jedes Ereignis (Start- bzw. Ende einer Aktivität) vier Zeitwerte: der früheste Eintrittszeitpunkt im besten bzw. schlechtesten Fall und der späteste Eintrittszeitpunkt im besten bzw. schlechtesten Fall. Diese entsprechen dem kürzesten bzw. längsten Instanztyp der zuvor diskutierten Ansätze. Die von PERT ebenfalls verwendete Varianz wird weitestgehend ignoriert.

Die Berechnung der vier Eintrittszeitpunkte findet auf Basis des Kontrollflusses und eventueller Minimal-Abstände statt. Maximal-Abstände werden in einem weiteren Schritt auf ihre Gültigkeit verifiziert, wobei lediglich der ermittelte Mittelwert berücksichtigt wird.

In [Pozewaunig, 1996] wird darüber hinaus eine Möglichkeit zur Berechnung von Schleifen durch das Schätzen der minimalen, maximalen und häufigsten Anzahl an Iterationen diskutiert. Aus diesen wird der Mittelwert und die Varianz der Anzahl der Iterationen auf ähnliche Weise wie bei den Dauern der Aktivitäten berechnet. Diese

können dann an geeigneter Stelle zur Schätzung der Gesamtlaufzeit der Schleife eingesetzt werden.

#### 3.1.1.3 MPM (Metra Potential Method)

Im Gegensatz zu den beiden bisher vorgestellten Netzplantechniken beruht die *Metra Potential Methode* [Kerbosh & Schell, 1975] auf *Vorgangs-Knoten-Netzen*. Dabei werden, wie in Abbildung 3.4 zu sehen, Aktivitäten als Knoten und die zeitlichen Abstände zwischen diesen als Kanten dargestellt.

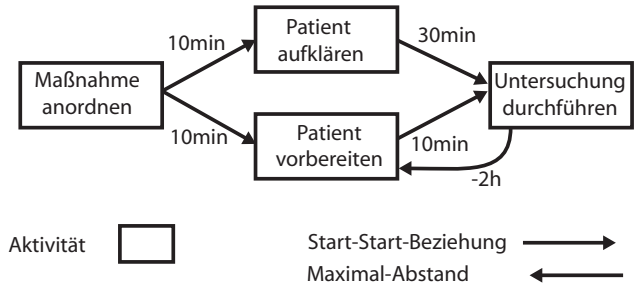



Abbildung 3.4: Vorgangs-Knoten-Netz

Diese Art der Modellierung stellt immer eine Start-Start-Abstandsbeziehung dar. Durch eine Aufsplittung der Aktivitäten in einen Start- und einen End-Vorgang lassen sich jedoch auch die übrigen drei Abstandsbeziehungen darstellen. Darüber hinaus lässt sich bei MPM-Graphen, neben einem Minimal-Abstand, durch das Einfügen einer zusätzlichen Kanten mit negativem Wert, ein Maximal-Abstand zwischen dem Ziel- und dem Quellknoten der Kante modellieren (siehe Abbildung 3.4). Eine Modellierung von Alternativ-Verzweigungen oder Schleifen ist mit der MPM nicht möglich.

Die Modellierung der Maximal-Abstände über Kanten mit negativem Wert führt dazu, dass ein MPM-Graph nicht mehr zyklenfrei ist (siehe Abbildung 3.4). Daher funktioniert der Basis-Algorithmus für MPM-Graphen nach dem *Label Correcting Verfahren* [Neumann & Morlock, 1993]. Berechnet werden für jede der Aktivitäten die vier Werte FAZ, SAZ, FEZ und SEZ (frühester, spätester Anfangszeitpunkt und frühester, spätester Endzeitpunkt). Dabei ist der Algorithmus auch in der Lage die Konsistenz bezüglich der gegebenen Maximal-Abstände zu überprüfen.

##### ADEPT-Time

ADEPT-Time ist eine in [Grimm, 1997] entwickelte Erweiterung des ADEPT-Modells um eine Zeitunterstützung auf Basis der MP-Methode. Hierbei werden die Aktivitäten, wie oben diskutiert, zunächst in Start- und Ende-Vorgänge aufgespaltet, wodurch eine

Modellierung aller vier Abstandsbeziehungen möglich ist. Zeitbedingungen zwischen den Vorgängen werden als Intervalle angegeben. Dabei wird die Unter- bzw. Obergrenze eines Intervalls als Minimal- respektive Maximal-Abstand betrachtet.

Eine Besonderheit in [Grimm, 1997] ist der Begriff der Schlüsselaktivitäten. Dabei handelt es sich um Aktivitäten mit zugewiesenen Fristen, welchen bei der Zeitberechnung eine besondere Rolle zukommt.

Zur Behandlung von Alternativ-Verzweigungen diskutiert [Grimm, 1997] einen optimistischen und einen pessimistischen Ansatz. Beim optimistischen Ansatz wird für die Berechnung der vier Zeitwerte jeweils angenommen, dass der Zweig mit den schwächsten zeitlichen Restriktionen durchlaufen wird. Der pessimistische Ansatz nimmt dagegen jeweils an, dass der Zweig mit den stärksten zeitlichen Restriktionen durchlaufen wird. Auch die Behandlung von Schleifen wird am Rande diskutiert. Dabei wird die Annahme gemacht, dass für die nachfolgenden Aktivitäten einer Schleife nur dann eine Zeitaussage bezüglich der spätesten Zeitpunkte getroffen werden kann, wenn entweder ein die Schleife umschließender Maximal-Abstand oder eine auf die Schleife folgende Schlüsselaktivität existiert. Ist dies nicht der Fall, wird die Maximal-Dauer der Schleife als $\infty$ angenommen. Es werden auch verschiedene Möglichkeiten zur Spezifikation von Zeitbedingungen innerhalb von Schleifen angegeben, die allerdings erst zur Laufzeit während der Ausführung der Schleife ausgewertet werden.

### 3.1.2 Temporal Constraint Network

Die *Temporal Constraint Networks* (TCN) gehören zur Klasse der *Constraint Satisfaction Probleme* (*CSP*) [Tsang, 1993], welche eine bekannte Problemstellung aus der KI darstellen. Es handelt sich dabei um eine in [Dechter *et al.*, 1991] vorgeschlagene Erweiterung der Standard-CSP um Variablen mit kontinuierlichem Wertebereich. *Temporal Constraint Networks* werden häufig als ein Netz von Knoten und Kanten dargestellt, dabei repräsentieren die Knoten Variablen und die Kanten Bedingungen, sogenannte (Binäre-)Constraints, welche an die Werte der beiden durch sie verbundenen Variablen geknüpft werden (siehe Abbildung 3.5). Die Knoten werden zusätzlich mit dem möglichen Wertebereich der Variable (Unäre-Constraints) und die Kanten mit der jeweiligen Bedingung beschriftet. Bei beidem handelt es sich um ein oder mehrere Intervalle, welche durch eine Vereinigung verknüpft sind.

Ziel eines *Temporal Constraint Networks* ist es, eine Belegung aller Variablen zu finden, sodass sämtliche Constraints erfüllt sind. Hierzu wird häufig zunächst ein sogenanntes minimales Netz berechnet, bei dem die Wertebereiche der Variablen bereits derart reduziert sind, dass jeder Wert Teil mindestens einer Lösung des Problems ist. Aufgrund dessen, dass TCNs mehrere durch Vereinigung verknüpfte Intervalle pro Constraint zulassen, besitzen ihre Algorithmen eine exponentielle Laufzeitkomplexität ($O(2^n)$). Eine Untergruppe der TCNs, für die dies nicht gilt, sind die *Simple Temporal Problems* (*STP*). Hier gilt gegenüber den normalen TCNs, dass die Constraints nur aus einem

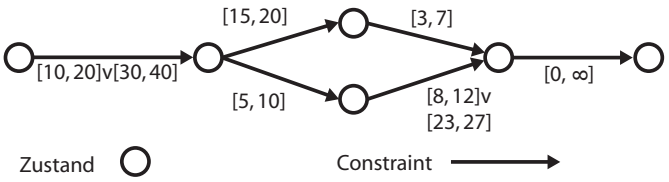



Abbildung 3.5: Darstellung eines TCN als Netz von Variablen und Constraints

einzelnen Intervall bestehen dürfen. Aufgrund dieser Einschränkung lässt sich das minimale Netz für ein STP beispielsweise durch einen Pfad-Konsistenz-Algorithmus wie den PC2-Algorithmus [Dechter *et al.*, 1991] in $O(n^3)$ bestimmen.

Eine interessante Eigenschaft der für STPs und TCNs verwendeten Algorithmen besteht darin, dass auch bisher unbekannte, implizit zwischen den Variablen geltende Bedingungen durch den Algorithmus bestimmt werden und Teil des minimalen Netzes sind. Hierdurch lassen sich auch bisher unbekannte Zusammenhänge bestimmen.

TCNs sind durch eine Aufsplittung der Aktivitäten in eine Variable für das Start- und eine für das End-Ereignis in der Lage, alle vier Abstandsbeziehungen zu repräsentieren. Es ist jedoch weder mit STPs noch mit TCNs möglich, die komplizierten Konstrukte Parallel-Verzweigung und Schleifen darzustellen.

#### Das Modell von Bettini, Wang und Jajodia

Eine Variante zur Zeitberechnung im Prozess-Management, die auf *Simple Temporal Problems* beruht, wurde in [Bettini *et al.*, 2002b] vorgestellt. Dabei werden die Aktivitäten in ein Start- und End-Ereignis aufgeteilt, womit eine Modellierung aller vier Abstandsbeziehungen möglich ist. Dauern und Abstände werden in Form von Intervallen angegeben, wobei das Modell von Bettini, Wang und Jajodia die Besonderheit besitzt, dass Intervalle unterschiedlicher Zeitgranularitäten unterstützt werden.

Zur Unterstützung von Alternativ-Verzweigungen werden die Prozessschemata an den XOR-Split-Knoten in ihre unterschiedlichen Instanztypen aufgeteilt und für jeden der Instanztypen ein eigenes STP erstellt. Danach wird für jedes der STPs sein minimales Netz sowie ein Schedule für die Start- und Endzeitpunkte der Aktivitäten berechnet. Die Schedules der verschiedenen Instanztypen werden schließlich – sofern möglich – zu einem Schedule vereinigt, welcher für das gesamte Prozessschema gilt. In [Bettini *et al.*, 2002b] werden dabei verschiedene Arten von Schedules diskutiert (free-, restricted-due-time-, bounded-Schedule), die danach unterschieden werden, ob und wie die Dauern der Aktivitäten eingeschränkt werden können.

### 3.1.3 Petri-Netze

Petri-Netze [Petri, 1962] sind ein weitverbreiteter Ansatz zur Modellierung, Simulation und Analyse nebenläufiger Systeme. Ein Petri-Netz ist ein bipartiter Graph mit zwei verschiedenen Typen von Knoten, bezeichnet als Stellen und Transitionen, die durch gerichtete Kanten verbunden sind. Jede Stelle ist in der Lage eine gewisse Menge von Markierungen (sog. Token) aufzunehmen, jeder Kante ist ein Gewicht zugeordnet. Eine Transition kann schalten, falls sich in allen eingehenden Stellen mindestens so viele Token befinden, wie die Transition Kosten verursacht. Schaltet eine Transition, so werden entsprechend Token aus den eingehenden Stellen gelöscht und entsprechend den Gewichten Token in die ausgehenden Stellen eingefügt.

Herkömmliche Petri-Netze werden auch zur Modellierung von Prozessschemata verwendet [Ellis & Nutt, 1993], [van der Aalst *et al.*, 1994], [van der Aalst *et al.*, 1998]. In ihrer reinen Form sind Petri-Netze zur Modellierung von Zeit nicht geeignet, es gibt jedoch verschiedene Erweiterung [Holliday & Vernon, 1985], [Tsai *et al.*, 1995], [Bause & Kritzinger, 1998], die eine Unterstützung zeitlicher Aspekte nachreichen. Petri-Netze sind dann in der Lage, sowohl alle Abstandsbeziehungen als auch alle Prozess-Konstrukte zu modellieren. Andererseits empfiehlt sich eine Modellierung von Alternativ-Verzweigungen und Schleifen – auch bei Petri-Netzen zur Modellierung von Prozessschemata – nur bedingt, da in diesem Fall die Eigenschaften des Systems nur noch durch Simulation analysiert werden können.

#### TCWF-net (Timing Constraint Workflow net)

Ein Beispiel für die Anwendung solcher erweiterter Petri-Netze für das Prozess-Management findet sich in [Li *et al.*, 2003]. Dabei wird, ausgehend von einem als gerichteter Graph modellierten Prozessschema, dieses zunächst in ein Petri-Netz transformiert, wie in Abbildung 3.6 zu sehen.

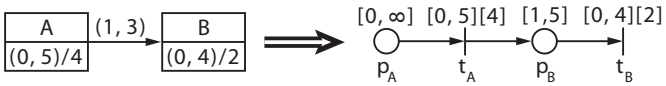



Abbildung 3.6: Ausschnitt eines Prozessschemas als TCWF [Li *et al.*, 2003]

Die Angaben aufseiten des Prozessschemas haben dabei folgende Bedeutung: $(0, 5)/4$ bei Aktivität A gibt an, dass die Aktivität 4 Zeiteinheiten dauert und zwischen 0 und 5 Zeiteinheiten nachdem sie aktiviert wurde (entspricht einem Wechsel von NOT_ACTIVATED nach ACTIVATED) ausgeführt werden kann, d. h. falls die Aktivität zum Zeitpunkt $T_0$ aktiviert wird, so ist ihr Ausführungszeitfenster $[T_0 + 0, T_0 + 5]$. Dasselbe gilt für $(0, 4)/2$ bei Aktivität B. Der Pfeil mit $(1, 3)$ zwischen den beiden Aktivitäten gibt deren Minimal-

und Maximal-Abstand an. Wird A zum Zeitpunkt $T_1$ beendet, so startet die Aktivität B im Zeitfenster $[T_1 + 1, T_1 + 3]$ und wird im Zeitfenster $[T_1 + 1, T_1 + 3 + 2]$ ausgeführt.

Bei der Transformation werden die Aktivitäten auf Transitionen abgebildet und entsprechende Stellen eingefügt. Die Zeitbedingung von A wird direkt auf die Zeitbedingung von $t_A$ übertragen. Dabei bedeutet $[0, 5]$, dass die Transition zwischen 0 und 5 Zeiteinheiten nach dem Eintreffen des Tokens in der vorhergehenden Stelle schaltet und für diesen Vorgang 4 Zeiteinheiten benötigt. Da B für die Ausführung 2 Zeiteinheiten benötigt und der Abstand zwischen A und B zwischen 1 und 3 Zeiteinheiten beträgt, bleibt das Token zwischen 1 und 5 Zeiteinheiten in Stelle $p_B$, bevor Transition $t_B$ schaltet. Parallel-Verzweigungen werden bei der Transformation durch Transitionen mit mehreren nachfolgenden Stellen und Alternativ-Verzweigung durch Stellen mit mehreren nachfolgenden Transitionen dargestellt.

Anhand dieser zeitlichen Modellierung eines Prozessschemas wird überprüft, ob es für jede Transition einen Zeitpunkt gibt, zu dem diese schaltbar ist. Schaltzeitpunkte für die Transitionen und damit Ausführungszeitpunkte für die Aktivitäten werden dabei nicht berechnet. Daneben wird durch Simulation ermittelt, ob sich durch eine erhöhte Anzahl gleichzeitiger Ausführungen eines Prozessschemas ein Engpass bei einer der Aktivitäten ergibt.

### 3.1.4 Weitere Formalismen

In diesem Abschnitt finden zwei weitere Formalismen zur Modellierung und Analyse von Zeit im Prozess-Management Erwähnung. Zum einen das auf der Intervall-Algebra von Allen beruhende T-WfMC-Modell aus [Kafeza & Karlapalem, 2000] und zum anderen der auf Timed-Automata beruhende Ansatz aus [Maria *et al.*, 2006].

#### T-WfMC

Bei T-WfMC [Kafeza & Karlapalem, 2000] handelt es sich um eine temporale Erweiterung des Prozess-Metamodells der *Workflow Management Coalition* (WfMC). Dabei wird die Intervall-Algebra von Allen [Allen, 1983] zur Modellierung des zeitlichen Ablaufs der Prozesse verwendet (siehe Abbildung 3.8) [WfMC, 1995]. Die Intervall-Algebra kennt dreizehn qualitative zeitliche Beziehungen der Art A vor B, A nach B, A startet B, A während B, etc. (siehe Abbildung 3.7). Die Dauern der Aktivitäten werden jeweils als ein Wert angegeben, wobei es keine Rolle spielt, ob dies der Maximal-, Minimal- oder ein Mittelwert ist.

Anhand der modellierten Zeitbeziehungen wird überprüft, ob alle Bedingungen ohne Einschränkung der Dauer eingehalten werden können. Daneben werden früheste Start- und späteste Endzeitpunkte für die Aktivitäten ermittelt. Zusätzlich ist es möglich, Fristen für die Aktivitäten anzugeben, deren Einhaltung der Algorithmus überprüft. Sowohl beim Überprüfen der Zeitbedingungen als auch beim Berechnen der Zeitpunkte werden Alternativ-Verzweigungen auf dieselbe Weise wie Parallel-Verzweigungen behandelt.

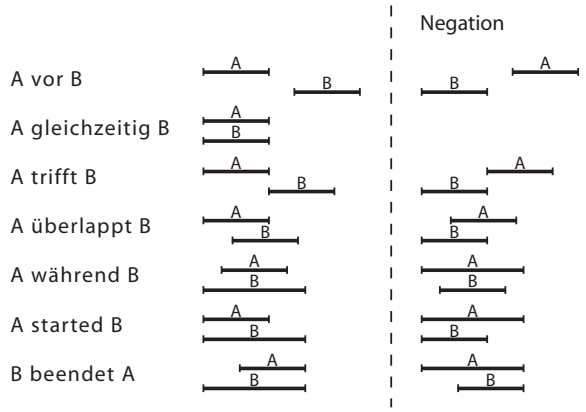



Abbildung 3.7: Intervall-Algebra [Allen, 1983]

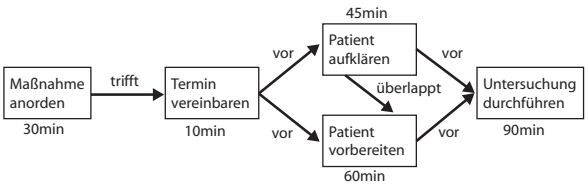



Abbildung 3.8: Prozessschema in T-WfMC

### Timed-Automata von Maria, Montanari und Zantoni

In [Maria *et al.*, 2006] wird die Anwendbarkeit von Timed-Automata [Alur & Dill, 1994] für die Modellierung und Verifikation der Zeitbedingungen von Prozessen untersucht. Timed-Automata stellen eine Erweiterung der endlichen Automaten [Stucky, 1995] um das Konzept von Uhren zur Darstellung von Zeit dar. Es ist mit ihnen möglich, sowohl Alternativ-Verzweigungen als auch Schleifen zu modellieren. Die Existenz von Parallel-Verzweigungen führt jedoch zu einem nicht-deterministischen Automaten [Stucky, 1995], da alle möglichen Ausführungsreihenfolgen einzeln betrachtet werden müssen.

Bei der Transformation eines Prozessschemas werden die einzelnen Prozess-Konstrukte zunächst in verschiedene Operationen eines Timed-Automata übersetzt. Unterstützt werden dabei ein fester Wert für die Dauer einer Aktivität sowie Minimal- und Maximal-Abstände zwischen den einzelnen Aktivitäten.

Die Analyse des Timed-Automata beruht darauf festzustellen, ob die von diesem akzeptierte Sprache leer ist oder nicht. Es wird somit ermittelt, ob es Wörter, das heißt eine Menge von Zeitpunkten, gibt, welche vom Automaten akzeptiert werden. Dies entspricht der Überprüfung, ob ein Lösungspfad mit entsprechenden Zeitpunkten

durch das Prozessschema existiert. Die Korrektheit aller möglichen Pfade ist damit nicht gewährleistet. Eine explizite Berechnung der Zeitpunkte findet ebenfalls nicht statt.

### 3.1.5 Kritischer Vergleich

Im folgenden Abschnitt werden die diskutierten Verfahren hinsichtlich ihrer Unterstützung für das Zeitmanagement in einem Prozess-Management-System miteinander verglichen. Dabei gibt Tabelle 3.1 zunächst einen Überblick über die wichtigsten Eigenschaften der vorgestellten Formalismen.

| | CPM | PERT | MPM | TCN | Petri-Netz |
|---|---|---|---|---|---|
| **Zeit** | | | | | |
| Zeitdauern/-abstände | 1 Wert | 3 Werte | 1 Wert | Intervall | Intervall |
| **Abstandsbeziehungen** | | | | | |
| Start-Start | ○ | ○ | ✓ | ✓ | ✓ |
| Start-Ende | ✗ | ✗ | ○ | ✓ | ✓ |
| Ende-Start | ✓ | ✓ | ○ | ✓ | ✓ |
| Ende-Ende | ✗ | ✗ | ○ | ✓ | ✓ |
| **Abstände** | | | | | |
| Minimal-Abstand | ✓ | ✓ | ✓ | ✓ | ✓ |
| Maximal-Abstand | ✗ | ✗ | ✓ | ✓ | ✓ |
| **Prozess-Konstrukte** | | | | | |
| Sequenz | ✓ | ✓ | ✓ | ✓ | ✓ |
| Parallel-Verzweigung | ✓ | ✓ | ✓ | ✓ | ✓ |
| Alternativ-Verzweigung | ○ | ○ | ○ | ○ | ○ |
| Schleifen | ✗ | ✗ | ✗ | ✗ | ✗ |
| **Berechnungspotenzial** | | | | | |
| Ausführbarkeit | ✓ | ✓ | ✓ | ✓ | ✓ |
| Startzeitpunkte | ✓ | ○ | ✓ | ✓ | ✗ |
| Implizite Bedingungen | ✗ | ✗ | ✗ | ✓ | ✗ |
| Komplexität[a] | $O(n^2)$ | $O(n)$ | $O(n^2)$ | $O(n^3)$ | $O(n^3)$ |

✓: vollständig unterstützt
○: bedingt unterstützt
✗: nicht unterstützt

[a] Abhängig von der Art des Verfahrens / der Modellierung

Tabelle 3.1: Überblick über Modelle zur Repräsentation von Zeit

CPM, MPM, T-WfMC und Timed-Automata unterstützen einen festen Wert für die Modellierung der Dauer einer Aktivität. Dabei ist jedoch nicht festgelegt, ob es sich um die minimale, maximale oder mittlere Dauer handelt. PERT stellt an dieser Stelle eine

Besonderheit dar. Hier werden drei Werte für die Dauern der Aktivitäten angegeben, die jedoch unter Verwendung einer stochastischen Funktion auf den Erwartungswert und die Varianz abgebildet werden. Temporal Constraint Networks und auch einige Varianten von Petri-Netzen verwenden Intervalle. Dabei ist es ohne Weiteres möglich, TCNs um eine Verwendung von Zeit-Histogrammen zu erweitern. Sowohl TCNs als auch Petri-Netze gewährleisten jedoch nicht, dass das angegebene Intervall vollständig zur Verfügung steht. Es wird lediglich sichergestellt, dass eine mögliche Lösung innerhalb der gegebenen Intervalle existiert.

MPM, TCN und Petri-Netze erlauben sowohl die Verwendung aller vier Abstandsbeziehung als auch die Verwendung von Minimal- und Maximal-Abständen zur Modellierung der zeitlichen Bedingungen. In diesem Punkt weisen CPM, PERT, T-WfMC und Timed-Automata einige Schwächen auf.

Petri-Netze und ebenso die Timed-Automata beschränken sich darauf, die Ausführbarkeit der Prozessschemata unter den gegebenen Zeitbedingungen zu überprüfen. Es werden jedoch keine möglichen Zeitpunkte für die Aktivitäten bestimmt, welche für ein ausgefeilteres Zeitmanagement von großem Interesse sind.

Die Timed-Automata sind zwar in der Lage alle Prozess-Konstrukte zu beschreiben, sie besitzen jedoch das Problem, dass für die entstehenden nicht-deterministischen Automaten nur NP-vollständige Algorithmen existieren. Dies macht sie für ein effizientes Prozess-Management ungeeignet.

Der Ansatz von Kafeza und Karlapalem besitzt einige interessante Eigenschaften, jedoch reichen die rein qualitativen Beziehungen nicht zur Beschreibung komplexer Zeitabhängigkeiten aus. Darüber hinaus ist es möglich, das Modell vollständig durch Temporal Constraint Networks zu modellieren und kann somit als ein Spezialfall dieser aufgefasst werden.

Die stochastischen Werte, die PERT für Dauern verwendet, erlauben keine eindeutigen zeitlichen Aussagen, welche aber für eine genauere Analyse und die Überwachung von Zeitbedingungen nötig sind. Auch die festen Werte, die von CPM und MPM ermittelt werden, erschweren eine Überwachung der Zeitbedingungen. Es bietet sich hier jedoch die Möglichkeit, die Berechnung sowohl für die Minimal- als auch die Maximal-Werte durchzuführen, um Unter- und Obergrenzen für die Ausführungszeitfenster zu erhalten. Für eine weitere Verwendung der berechneten Zeitdaten, beispielsweise im Rahmen eines Ressourcenmanagements, sind Ausführungszeitfenster für die einzelnen Aktivitäten unerlässlich. Daher scheidet PERT an dieser Stelle ebenfalls aus.

Fast alle der vorgestellten Ansätze haben erhebliche Probleme mit Alternativ-Verzweigungen und Schleifen. Dies ist jedoch auch kein triviales Problem, welches schon aufgrund seiner Struktur zu Algorithmen mit hoher Laufzeitkomplexität führt. Es gibt jedoch mehrere Ansätze, die durch verschiedene Einschränkungen in der Lage sind, effizient mit diesem Problem umzugehen (siehe Abschnitt 4.1.6).

Bisher bieten MPM und TCN die beste Unterstützung der geforderten Eigenschaften. Ein weiteres Kriterium, welches für die Verwendung von TCNs spricht, ist deren Mög-

lichkeit implizite Zeitbedingungen zwischen Aktivitäten zu ermitteln. Dies ist vor allem bei der späteren Analyse der berechneten Zeitdaten von großem Interesse. Die erhöhte Laufzeitkomplexität gegenüber MPM ($O(n^2)$ vs. $O(n^3)$) kann an dieser Stelle ignoriert werden.

Die in diesem Abschnitt beschriebenen Verfahren weisen für eine adäquate Zeitmodellierung von Prozessen eine Reihe von Schwächen auf. Über das hier diskutierte hinaus gibt es noch einige weitere Kritikpunkte wie beispielsweise, dass keines der Verfahren dynamische Änderungen zur Laufzeit berücksichtigt. Für weiterführende Überlegungen können die vorgestellten Verfahren jedoch als Grundlage dienen. Besonders TCNs bieten sich hierfür an: Sie sind zwar nur fähig die beiden Prozess-Konstrukte Sequenz und Parallel-Verzweigung abzubilden, jedoch verfügen sie über eine realistische Darstellung der Dauer als Intervalle sowie über die Möglichkeit zur Modellierung aller vier Abstandsbeziehungen. Außerdem sind sie in der Lage, *Ausführungszeitfenster* für die Aktivitäten eines Prozessschemas zu berechnen, welche für eine weitere Verwendung des Zeitmanagements, über die reine Überwachung der Zeitbedingungen hinaus, wichtig ist. Auf dieser Basis kann das Verfahren um eine Unterstützung der weiteren Konstrukte und anderer Eigenschaften erweitert werden. Dabei muss auch auf die Frage der Sicherstellung der Konsistenz näher eingegangen werden.

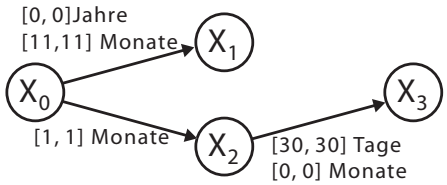
## 3.2 Weiterführende Konzepte

Das Zeitmanagement in einem Prozess-Management-System endet nicht bei der Modellierung und Überprüfung der Zeiteigenschaften der Prozesse. Wichtig ist etwa auch, wie in Kapitel 2 diskutiert, wie mit unterschiedlichen Zeiteinheiten verfahren werden kann, was im Fall eines Zeitfehlers zu geschehen hat oder wie man versuchen kann, diese weitestgehend zu vermeiden. Einige in der Literatur vorgestellte Ansätze zur Lösung dieser und anderer Problemstellungen werden in diesem Abschnitt kurz vorgestellt.

*Zeiteinheiten* sind ein wichtiges Thema bei der Unterstützung von Zeit im Prozess-Management. Einige der dabei auftretenden Probleme wurden bereits in Abschnitt 2.2.3 diskutiert. Es gibt bisher nur wenige Ansätze [Bettini *et al.*, 2002b], [Combi *et al.*, 2004], [Goralwalla *et al.*, 2001] zur Behandlung verschiedener Zeiteinheiten jedoch ist mit keinem dieser Ansätze ein Rechnen auf Basis der Zeiteinheiten möglich. Vielmehr ist es nötig, die Zeitbedingungen eines Zeitgraphen zu analysieren, um Inkonsistenzen zu erkennen. Beispiel 3.3 verdeutlicht einige der hierfür verantwortlichen Eigenschaften von Zeiteinheiten. In [Bettini *et al.*, 1998] wird darüber hinaus gezeigt, dass die Algorithmen für die Bestimmung der Konsistenz eines *Temporal Constraint Networks* mit beliebigen Zeiteinheiten NP-vollständig sind.

Bezüglich des Auftretens von Zeitgranularitäten in Kombination mit Fristen (z. B. „jeder dritte Montag“), gibt es neben einer vollständigen Unterstützung auch weitere Möglichkeiten diese zu behandeln. In [Eder *et al.*, 1999a] wird beispielsweise eine Trans-

Die Inkonsistenz des nachfolgenden Netzwerks kann nicht durch einen Algorithmus erkannt werden, welcher auf Basis der Manipulation von Bedingungen operiert.




Die Schwierigkeit ist, das der Algorithmus erkennen muss, das der Wertebereich von $X_2$ auf Februar eingeschränkt ist, da die Bedingung zwischen $X_0$ und $X_1$ den Wert von $X_0$ auf Januar einschränkt (Werte von $X_0$ müssen im selben Jahr wie $X_1$ liegen nur 11 Monate früher) und die Bedingung zwischen $X_0$ und $X_2$ den Wert von $X_2$ auf einen Monat später festlegt. Da die Bedingung zwischen $X_2$ und $X_3$ jedoch besagt, dass die beiden im selben Monat (Februar), aber 30 Tage voneinander entfernt liegen müssen, ist das Netzwerk inkonsistent, da dies aufgrund der Eigenschaften des Monats Februar nicht möglich ist.

Beispiel 3.3: Rechnen mit Zeiteinheiten [Bettini *et al.*, 2002a]

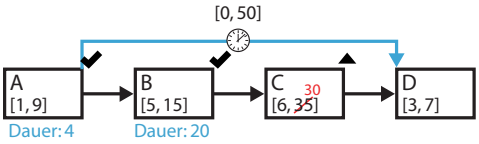
formation solcher Fristen in Mindest-Abstände mithilfe einer Worst-Case-Abschätzung diskutiert. Dadurch kann garantiert werden, dass ein, die Bedingungen erfüllender Prozess, auch entsprechend der Fristen ausgeführt werden kann. Hierbei lassen sich Abhängigkeiten zwischen verschiedenen Fristen ausnutzen, eine Idee die in [Chen *et al.*, 2004] und [Chen & Yang, 2004] verfolgt wird. Die dort angestrengten Überlegungen zu Abhängigkeiten verschiedener Zeitbedingungen erlauben darüber hinaus eine Optimierung der Konsistenzprüfung.

[Zhuge *et al.*, 2001] untersuchen, welche Optimierungsmöglichkeiten und Probleme sich bei der über mehrere Zeitzonen verteilten Ausführung eines Prozesses ergeben. Dabei müssen unter anderem die unterschiedlichen Arbeitszeiten in den einzelnen Zeitzonen beachtet werden. Außerdem kann die Zeitdauer des Datenflusses nicht mehr vernachlässigen werden, da der Transfer von Ressourcen über Kontinente hinweg möglicherweise einige Zeit benötigt.

Treten Zeitverzögerungen auf, so sind manche Bedingungen nicht mehr zu halten und die Prozessinstanz wird inkonsistent. Damit muss ein Zeitfehler ausgelöst werden, was zu hohen Kosten führen kann. Sowohl [Eder & Panagos, 2000] als auch [Chen & Yang, 2005b] untersuchen daher weiter abgestufte Fehlerzustände. Sie betrachten hierzu zunächst, wie hoch die Wahrscheinlichkeit ist, dass die Zeitverzögerung durch die eventuell zu großzügig geplanten Dauern anderer Aktivitäten ausgeglichen wird. Besteht die Möglichkeit die Zeitverzögerung auszugleichen, schränken sie dementsprechend die

Dauern der nachfolgenden Aktivitäten ein und erhalten damit wieder einen konsistenten Zustand (siehe Beispiel 3.4).

Dauert bei der dargestellten Prozessinstanz Aktivität B länger als 15 Zeiteinheiten, so muss ein Zeitfehler ausgelöst werden, da der Maximal-Abstand von 50 Zeiteinheiten zwischen A und D nicht mehr eingehalten werden kann.




Aufgrund der hohen Differenz zwischen minimaler und maximaler Dauer der Aktivität C kann jedoch davon ausgegangen werden, dass diese im Normalfall nicht die erlaubten 35 Zeiteinheiten benötigt. Es ist daher möglich die Maximal-Dauer der Aktivität C auf 30 Zeiteinheiten zu reduzieren, wodurch auch der Maximal-Abstand zwischen A und D wieder eingehalten werden kann.

Beispiel 3.4: Korrektur einer Zeitverzögerung durch Einschränkung der Dauern

Ein weiterer interessanter Vorschlag in [Pozewaunig, 1996] im Zusammenhang mit Zeitverzögerungen ist es, ab einem bestimmten Zeitpunkt, das heißt einer gewissen Verspätung, lange Pfade einer Alternativ-Verzweigung nicht mehr zuzulassen, um dadurch die Verspätung zu reduzieren und so das Auslösen eines Zeitfehlers zu vermeiden. Eine Weiterführung dieses Gedankens findet sich in [Eder *et al.*, 1999b], hier wird vorgeschlagen, gewisse Aktivitäten als optional zu markieren. Diese können dann vom Prozess-Management-System ausgelassen werden, falls dies zur Vermeidung von Zeitfehlern hilfreich ist.

Für ein adaptives Prozess-Management, bei dem Prozessinstanzen zur Laufzeit geändert werden können, ist es wichtig, den Einfluss solcher Änderungen auf das Zeitmanagement zu bestimmen. In[Sadiq *et al.*, 2000] wird hierfür eine 3-Phasen-Modifikation empfohlen: Zuerst werden die nötigen Modifikationen bestimmt, danach wird überprüft auf welche der Prozessinstanzen diese anwendbar sind und im dritten Schritt werden die entsprechenden Instanzen modifiziert.

## 3.3 Angrenzende Themenbereiche

Das Zeitmanagement stellt nur einen kleinen Teil eines Prozess-Management-Systems dar. Es gibt noch viele weitere Aufgaben eines solchen Systems, welche vom Zeitmanagement vollständig unabhängig sind, wie beispielsweise die Verwaltung der anfallenden Daten oder die Integration der organisatorischen Strukturen. Es gibt jedoch auch einige mögliche

Aufgaben eines solchen Systems, die eng mit dem Zeitmanagement verbunden sind oder sogar davon abhängig sind.

Ein Beispiel für ein solches Aufgabengebiet ist die Sicherstellung vonseiten des PMS, dass die Aktivitäten in den ihnen vorgegebenen Zeitintervallen ausgeführt werden. Da vor allem ein menschlicher Benutzer aber nicht zum Ausführen einer Aktivität zu einem gewissen Zeitpunkt gezwungen werden kann, müssen hier alternative Wege beschritten werden. [Zhao & Stohr, 1999] favorisiert hierzu eine dynamische Priorisierung und Distribution der Arbeitseinheiten anhand verschiedener Strategien, wie Shortest-Remaining-Time[3] oder Weighted-Remaining-Time[4]. Dabei soll erreicht werden, dass sämtliche Instanzen eines Prozesses ähnliche Durchlaufzeiten besitzen, um beispielsweise bei der Bearbeitung von Reklamationen zuverlässige Aussagen über die Bearbeitungsdauer treffen zu können.

[Eder *et al.*, 2003] hingegen bevorzugen sogenannte *Personal Schedules*, dabei wird für jeden Benutzer ein eigener Terminplan erstellt. Dieser garantiert, dass unter normalen Bedingungen und sofern sich der Benutzer daran hält, alle Aktivitäten in den für sie vorgesehenen Zeitintervallen ausgeführt werden. Dabei werden bei der Erstellung des Personal Schedules auch Aktivitäten beachtet und für die Zukunft eingeplant, welche aufgrund des Zustands des aktuellen Kontrollflusses erst in naher Zukunft gestartet werden können. Dies ermöglicht dem Benutzer auch eine gewisse Vorausschau auf zukünftige Arbeitsschritte.

Lässt sich eine Zeitüberschreitung nicht mehr vermeiden oder ist die Wahrscheinlichkeit dafür sehr hoch, so kann es kostengünstiger sein, eher früher als später einen Zeitfehler auszulösen. Welche Bedingungen hierbei für eine Kostenersparnis erfüllt sein müssen und zu welchem Zeitpunkt der Zeitfehler schlussendlich ausgelöst werden sollte wird in [Panagos & Rabinovich, 1997] diskutiert.

Im Zusammenspiel mit Fristen der Art „jeder dritte Montag des Monats“ kann es schon bei einer etwas zu späten Bereitstellung einer Aktivität zu erheblichen Verzögerungen kommen. Steht beispielsweise bei einer solchen Zeitangabe die Aktivität erst einen Tag später bereit, muss mindestens 4 Wochen auf eine erneute Möglichkeit zur Ausführung gewartet werden. Wie solche unerwünschten Effekte berechnet und deren Auswirkungen abgeschätzt werden können wird in [Eder *et al.*, 2005] untersucht.

Einen ähnlichen Charakter haben die in [Zerguini, 2001] und [Li *et al.*, 2003] untersuchten Möglichkeiten zur Abschätzung von Durchlaufzeiten unter Last. Dabei wird ermittelt, wie sich die Durchlaufzeiten der Instanzen eines einzelnen Prozesses bei verschieden vielen gleichzeitig ablaufenden Prozessinstanzen verhalten.

Eine weitere Aufgabe von PMS ist das Ressourcenmanagement. Dabei ist das Ziel einerseits eine Verzögerung von Aktivitäten durch benötigte, aber momentan nicht zur Verfügung stehende Ressourcen, zu vermeiden, andererseits soll aber auch eine möglichst gute Auslastung der zur Verfügung stehenden Ressourcen erreicht werden.

[3] engl.: kürzeste verbleibende Zeit
[4] engl.: gewichtete verbleibende Zeit

Eine Möglichkeit, wie mittels erwarteter Ausführungszeitfenster an die Realisierung einer solchen Funktion herangegangen werden kann, wird in [Li & Yang, 2005] diskutiert. Dazu wird zunächst, über alle aktiven Prozessinstanzen hinweg, mithilfe der erwarteten Ausführungszeitfenster der Aktivitäten, ermittelt, welche Paare von Aktivitäten mögliche Ressourcenkonflikte aufweisen. Darauf aufbauend werden die konfligierenden Aktivitäten-Paare – auch über verschiedene Prozessinstanzen hinweg – mittels einer First-Come-First-Served-Strategie[5] zeitlich so gegeneinander angeordnet, dass keine Ressourcenkonflikte mehr bestehen. Dazu wird das Ausführungszeitfenster der später kommenden Aktivitäten des Paares derart nach hinten verschoben, dass keine zeitliche Überschneidung der beiden Aktivitäten mehr besteht. An dieser Stelle wird jedoch nicht weiter auf die Frage eingegangen, auf welche Weise, aus einer Verschiebung der Aktivitäten resultierende, Zeitüberschreitungen behandelt werden können.

## 3.4 Zusammenfassung

Im Zusammenhang mit Zeitaspekten in Informationssystemen im Allgemeinen und bei Prozess-Management-Systemen im Detail gibt es eine Vielzahl wissenschaftlicher Publikationen. In diesem Kapitel wurden einige dieser Arbeiten vorgestellt und kritisch betrachtet.

Zunächst haben wir einige Verfahren zur Repräsentation der Zeitbedingungen und zur Verifikation der dabei verwendeten Zeitmodelle untersucht. Besondere Aufmerksamkeit galt hierbei den in Kapitel 2 gestellten Anforderungen hinsichtlich der zu unterstützenden Prozess- und Zeitkonstrukte. Darüber hinaus wurde betrachtet, inwiefern die Verfahren in der Lage sind, die Ausführbarkeit der Zeitmodelle sicherzustellen und Ausführungszeitpunkte für die Aktivitäten zu ermitteln. Es stellte sich unter anderem heraus, dass Alternativ-Verzweigungen und Schleifen von den untersuchten Verfahren bisher kaum bzw. nur unzureichend unterstützt werden. Ein abschließender Vergleich der Verfahren ergab jedoch, dass einige als Grundlage für weiterführende Überlegungen geeignet sind, wobei besonders die *Temporal Constraint Networks* hervortraten.

Im Anschluss wurden einige Ansätze zu weiterführenden Fragestellungen des Zeitmanagements betrachtet. Hierzu zählen etwa die Behandlung unterschiedlicher Zeiteinheiten oder Möglichkeiten zur Vermeidung von Zeitfehlern. Bei der Behandlung unterschiedlicher Zeiteinheiten konnten wir feststellen, dass eine vollständige Unterstützung nicht empfehlenswert ist, da es sich hierbei um ein NP-vollständiges Problem handelt.

Abschließend haben wir an das Zeitmanagement angrenzende Themenbereiche untersucht, die insbesondere von einem funktionierenden Zeitmanagement abhängig sind. Zu diesen Themen gehören unter anderem das Ressourcenmanagement und persönliche Terminpläne für die einzelnen Benutzer. Im Vordergrund stand dabei, welche Informationen

[5] engl.: Den Ersten zuerst bedienen.

vom Zeitmanagement zur Unterstützung solcher Aspekte bereitgestellt werden müssen. Dies sind vor allem das Startzeitfenster und die Dauer einer jeden Aktivität.

Im folgenden Kapitel werden zunächst die für die Modellierung der Zeitbedingungen nötigen Erweiterungen des ADEPT2-Metamodells vorgestellt und erläutert. Danach wird ein Zeitmodell sowie darauf operierende Algorithmen angegeben. Ersteres ermöglicht eine einfachere Rechnung mit den für den Prozess spezifizierten Zeitbedingungen, da es das Prozessschema auf den für die Zeitrechnung wesentlichen Teil reduziert. Die Algorithmen sind dann in der Lage, aus dem Zeitmodell die Konsistenz des Prozessschemas sowie Zeitpunkte für die Aktivitäten abzuleiten.

# 4 Integration zeitlicher Aspekte in ein Prozess-Metamodell

Im Folgenden werden die in Kapitel 2 diskutierten Anforderung konkret in einem Prozess-Metamodell umgesetzt. Dies erfolgt beispielhaft am *ADEPT2-Basismodell*, wobei die notwendigen Erweiterungen als *ADEPT2-Time-Modell* mit einfließen. Das Prozess-Metamodell muss hierzu um Konstrukte erweitert werden, die dazu dienen, die zeitlichen Abhängigkeiten eines Prozesses zu modellieren. Außerdem müssen die bereits existierenden Konstrukte um die, für ihre korrekte Behandlung, nötigen Zeiteigenschaften erweitert werden. Eine wichtige Frage ist dabei auch, wie im Fall der Existenz verschiedener Instanztypen verfahren werden kann. Für dieses Problem werden verschiedene Lösungsansätze präsentiert und erörtert.

Im nächsten Schritt wird eine kompakte Darstellung für die Zeitbedingungen eines Prozessschemas entwickelt. Dieses *Zeitmodell* stellt ein Prozessschema in einer auf die Zeiteigenschaften reduzierten Form dar. Bei der anschließenden Entwicklung der Zeitmanagement-Algorithmen ist es dadurch möglich, für die Algorithmen unwichtige prozessspezifische Details zu ignorieren. Zusätzlich werden die impliziten Zeitinformationen der Elemente des Prozessschemas durch das Zeitmodell explizit dargestellt und können somit in den Algorithmen leichter verwendet werden. Anschließend wird eine Transformation benötigt, welche einen im ADEPT2-Time-Modell spezifizierten Prozess in das Zeitmodell übersetzt und es erlaubt, die dort gewonnen Ergebnisse zurück in das jeweilige Prozessschema zu übertragen. Auf Basis des Zeitmodells ist es dann möglich, Algorithmen anzugeben, welche zunächst dessen Konsistenz überprüfen und, sollte diese gegeben sein, mögliche Ausführungszeitpunkte für die Aktivitäten bestimmen.

Dieses Kapitel gliedert sich daher wie folgt: Im ersten Abschnitt wird das ADEPT2-Time-Modell mit seinen Erweiterungen gegenüber dem ursprünglichen ADEPT2-Basismodell beschrieben. Dabei wird sowohl Syntax als auch Semantik der neuen Zeiteigenschaften definiert und es wird ausführlich auf Möglichkeiten zur Behandlung verschiedener Instanztypen eingegangen.

Abschnitt 4.2 beschäftigt sich dann mit der Definition des erwähnten Zeitmodells sowie verschiedenen Algorithmen, welche auf diesem operieren. Es wird dabei beispielsweise auch erläutert, wie Inkonsistenzen in den Zeitbedingungen eines Prozessschemas entdeckt und lokalisiert werden können und welche verschiedenen Möglichkeiten es zur Berechnung der Ausführungsintervalle gibt.

Abgeschlossen wird das Kapitel durch eine kurze Zusammenfassung, die nochmals auf die wichtigsten Punkte eingeht und diese in einen gemeinsamen Kontext bringt.

## 4.1 ADEPT2-Time

Wie erwähnt stellt ADEPT2-Time eine Erweiterung des ADEPT2-Basismodells um Elemente für die Modellierung der zeitlichen Abhängigkeiten dar. Um die zeitlichen Eigenschaften der Knoten und Kanten spezifizieren zu können, müssen deren Definitionen um einige Zeitelemente erweitert werden. Darüber hinaus müssen zusätzliche Zeitkanten eingeführt werden, um auch in der Lage zu sein, komplexere zeitliche Abhängigkeiten anzugeben.

Für die Ausführung der Prozesse ist es nötig, einen Zeitzustand einzuführen, welcher weitgehend orthogonal zu den bereits existierenden Aktivitätenzuständen ist. Mit diesem kann überwacht werden, in wie fern die aktuellen Zeitpunkte der Aktivitäten den geforderten Werten entsprechen.

Zunächst sollen jedoch die Wertebereiche der verwendeten Zeitvariablen genauer beschrieben werden. Dafür muss geklärt werden, wie die in Abschnitt 2.2.3 diskutierten Zeiteinheiten zu behandeln sind.

### 4.1.1 Zeitpunkte und Zeitdauern

Ein *Zeitpunkt* ist definiert als ein festgelegter Punkt im Ablauf eines Prozesses, dessen Lage durch Zeiteinheiten beschrieben und auf einen Nullpunkt bezogen ist [DIN, 1987]. Die Menge der möglichen Zeitpunkte ist durch $\mathbf{TP}_{NP}$ gegeben, wobei $NP$ den Bezugspunkt, das heißt den Nullpunkt der Zeitpunkte angibt. In dieser Arbeit sind vor allem zwei Bezugspunkte von Interesse.

- Bei $\mathbf{TP}_{Cal}$ handelt es sich um die Zeitpunkte eines normalen Kalenders, wie zum Beispiel des gregorianischen Kalenders. Der dabei verwendete Kalender kann vom konkreten System abhängen, was für die folgenden Betrachtungen jedoch ohne Bedeutung ist.

- Bei $\mathbf{TP}_{Ws}$ handelt es sich um Zeitpunkte, die relativ zum Start der Prozessinstanzen sind, das heißt also, der Start-Zeitpunkt der Prozessinstanz dient als Bezugspunkt.

Für Zeitpunkte können eine ganze Reihe von Zeiteinheiten definiert werden. In dieser Arbeit beschränken wir uns jedoch auf eine Angabe in Form von Datum und Uhrzeit. Das heißt, Zeitpunkte werden in der Form „Tag.Monat.Jahr Stunde:Minute“ angegeben, wobei die jeweiligen Angaben relativ zum Nullpunkt des Bezugssystems sind. Ist der Datumsanteil eines Zeitpunktes leer, d. h. „00.00.0000“, so kann er auch weggelassen werden. Besondere Zeitpunkte sind $+\infty$ und $-\infty$, welche einen undefinierten Zeitpunkt repräsentiert.

Eine *Zeitdauer* ist die Zeitspanne zwischen zwei Ereignissen; das heißt, sie entspricht der Differenz zwischen den Zeitpunkten der beiden Ereignisse. Auch Zeitdauern werden in verschiedenen Einheiten angegeben. Die für uns relevanten sind Jahre, Monate, Wochen, Tage, Stunden und Minuten. Dabei sind auch konsistente Verknüpfungen (z. B. 1 Woche 3 Tage 7 Stunden) dieser Einheiten möglich. Die Menge der zulässigen Zeitdauern ist durch **TD** gegeben. Des weiteren gibt es zwei ausgezeichnete Zeitdauern, zum einen die Zeitdauer 0, bei der beide Ereignisse zum selben Zeitpunkt stattfinden, und zum anderen die Zeitdauer $\infty$, bei der die Zeitpunkte der beiden Ereignisse unabhängig voneinander sind.

Im Folgenden werden häufig Intervalle von Zeitdauern und Zeitpunkten verwendet. Für Zeitdauern hat ein solches Intervall $[\mathtt{D}^{min}, \mathtt{D}^{max}] \subseteq \mathbf{TD}$ die Bedeutung, dass zwischen den Zeitpunkten der beiden Ereignisse mindestens eine Zeitdauer von $\mathtt{D}^{min}$ und höchstens eine Zeitdauer von $\mathtt{D}^{max}$ vergehen darf. Für Zeitpunkte hat ein Intervall $[\mathtt{FZ}, \mathtt{SZ}] \subseteq \mathbf{TP}$ die ähnliche Bedeutung, dass das Ereignis frühestens zum Zeitpunkt FZ eintreten darf und spätestens zum Zeitpunkt SZ eingetreten sein muss.

Um Intervalle in Formeln leichter erkennen zu können, werden die eckigen Klammern zur Markierung verwendet. Dementsprechend bezeichnet [**TD**] ein Intervall von Zeitdauern und [**TP**] ein Intervall von Zeitpunkten.

Eine Zeitdauer kann auch als ein Zeitpunkt aufgefasst werden und umgekehrt, wobei das Ursprungsereignis der Zeitdauer als Bezugspunkt des Zeitpunktes fungiert. Das heißt für die Zeitdauer $d \in \mathbf{TD}$ zwischen zwei Ereignissen $E_i$ und $E_j$ gilt $d = p \in \mathbf{TP}_{E_i}$.

### 4.1.2 Erweiterung von Knoten

Die Definition der *Knoten* aus Abschnitt 2.1.3.2 wird zunächst um einen Zeitparameter $\mathtt{Time}_n$ zum 5-Tupel $(\mathtt{ID}_n, \mathtt{nT}_n, \mathbf{Pars}_n, \mathbf{M}_n, \mathtt{Time}_n)$ erweitert. Dabei ist

$$\mathtt{Time}_n =< [\mathtt{FZ}_n^{begin}, \mathtt{SZ}_n^{begin}], [\mathtt{FZ}_n^{end}, \mathtt{SZ}_n^{end}], [\mathtt{D}_n^{min}, \mathtt{D}_n^{max}], \mathtt{K}_n, \mathtt{P}_n > .$$

Weiter sind $[\mathtt{FZ}_n^{begin}, \mathtt{SZ}_n^{begin}], [\mathtt{FZ}_n^{end}, \mathtt{SZ}_n^{end}] \subseteq \mathbf{TP}_{Ws}$ Intervalle von Zeitpunkten, welche das Start- und Endzeitintervall, das heißt den frühesten und spätesten Startzeitpunkt und den frühesten und spätesten Endzeitpunkt des Knotens repräsentieren. Das Intervall $[\mathtt{D}_n^{min}, \mathtt{D}_n^{max}] \subseteq \mathbf{TD}$ gibt die minimale und maximale Dauer des Knotens an. Dieser Wert kann meist der dem Knoten zugrunde liegenden Aktivität entnommen werden, wobei sein Standardwert $[0, \infty]$ ist. $\mathtt{K}_n$ ist eine Funktion, die, sofern gegeben (d. h. $\mathtt{K}_n \neq \varnothing$), angibt, dass es sich bei diesem Knoten um eine Schlüsselaktivität (siehe Abschnitt 4.1.2.1) handelt. Gleichfalls handelt es sich bei $\mathtt{P}_n$ um eine Funktion, die bei Aktivitäten mit dynamischer Dauer verwendet wird (siehe Abschnitt 4.1.2.2).

#### 4.1.2.1 Schlüsselaktivitäten

Eine *Schlüsselaktivität* ist eine spezielle Aktivität, deren Zeitdaten ($[\mathtt{FZ}_n^{begin}, \mathtt{SZ}_n^{begin}]$, etc.) nicht durch das Zeitmanagement-System aus Zeitdaten anderer Aktivitäten berechnet,

sondern von einer externen Komponente bereitgestellt werden. Diese Zeitdaten können aus verschiedenen Quellen stammen, zum Beispiel können sie

- von einem anderen System übernommen,
- aus einem Kalender abgeleitet,
- während des Prozesses in einem anderen Schritt ermittelt oder
- bereits auf Modellierungsebene festgelegt worden sein.

Das eine Schlüsselaktivität auszeichnende $\mathtt{K}_n$ ist eine Funktion

$$f : [\mathbf{TP}] \times [\mathbf{TP}] \rightarrow [\mathbf{TP}] \times [\mathbf{TP}]$$

mit deren Hilfe das Start- und Endzeitintervall der Aktivität zu gegebener Zeit bestimmt werden kann. Werden die entsprechenden Intervalle zur Laufzeit bestimmt, so muss sich die Funktion bei der Wahl des Termins an den durch die entsprechenden Intervalle $[\mathtt{FZ}_n^{begin}, \mathtt{SZ}_n^{begin}]$ beziehungsweise $[\mathtt{FZ}_n^{end}, \mathtt{SZ}_n^{end}]$ bereits vorgegebenen Rahmen halten. Das heißt, die Funktion darf die Werte dieser beiden Intervalle lediglich einschränken. Damit die Funktion in der Lage ist diese Bedingung zu erfüllen, wird sie auf die aktuellen Werte für $[\mathtt{FZ}_n^{begin}, \mathtt{SZ}_n^{begin}]$ und $[\mathtt{FZ}_n^{end}, \mathtt{SZ}_n^{end}]$ angewendet. Der Funktionswert von $f$ muss dann Teilmenge des Funktionsarguments sein, d. h. es muss gelten:

$$\forall [a], [b] \in [\mathbf{TP}] : f([a], [b]) = ([x], [y]) \Rightarrow ([a] \subseteq [x] \wedge [b] \subseteq [y]).$$

Diese Einschränkung beruht auf dem Umstand, dass diese Wertebereiche durch weitere Bedingungen innerhalb des Prozessschemas entstanden sind und ein Nichteinhalten zu einer Inkonsistenz gegenüber den entsprechenden Bedingungen führen würde.

Da Schlüsselaktivitäten eine wichtige Stellung bei der Modellierung von Prozessen und im Zeitmanagement zukommt, werden sie durch ein Uhr-Symbol, in der rechten oberen Ecke des Aktivitäten-Symbols, gekennzeichnet (siehe Abbildung 4.1). Die besondere Stellung der Schlüsselaktivitäten gründet sich auf der Eigenschaft, dass möglicherweise die Zeitdaten anderer Aktivitäten von den Zeitdaten einer Schlüsselaktivität abhängig sind. Hat eine Schlüsselaktivität beispielsweise einen festen Termin wie den 01.01.2009 für ihren spätesten Startzeitpunkt, so ist klar, dass alle der Schlüsselaktivität vorangehenden Aktivitäten bis zu diesem Termin abgeschlossen sein müssen. Damit muss, sobald der Termin einer Schlüsselaktivität feststeht, eventuell die Terminierung abhängiger Aktivitäten neu ermittelt werden.

#### 4.1.2.2 Aktivitäten mit dynamischen Dauern

Bei Aktivitäten mit *dynamischen Dauern* wird die minimale und maximale Dauer der Aktivität zur Laufzeit festgelegt. Dazu handelt es sich bei $\mathtt{P}_n$ sofern gegeben, um eine Funktion

$$g : [\mathbf{TD}] \rightarrow [\mathbf{TD}],$$


Abbildung 4.1: Graphische Darstellung einer Schlüsselaktivität

mit welcher die Dauer zu gegebener Zeit bestimmt werden kann. Die konkrete Funktion könnte beispielsweise ein Verweis auf ein, von einer anderen Aktivität geschriebenes, Datenelement oder auf eine externe Komponente darstellen. Dabei gilt auch hier, dass bei der Festlegung der Dauer auf den durch $[\mathtt{D}_n^{min}, \mathtt{D}_n^{max}]$ vorgegeben Rahmen geachtet werden muss. Es ist lediglich erlaubt diesen weiter einzuschränken, eine Erweiterung hingegen ist nicht möglich. Um dies einhalten zu können, wird die Funktion auf das Intervall angewendet, welches den Rahmen vorgibt. Es gilt somit

$$\forall [x] \in [\mathbf{TD}] : g([x]) = [y] \subseteq [x].$$

Für dynamische Dauern gibt es verschiedene Einsatzszenarien. Beispielsweise ist die Lieferdauer einer Bestellung meist erst zum Zeitpunkt der Abgabe der Bestellung bekannt. Da hier andererseits je nach Verfügbarkeit große Unterschiede zwischen den einzelnen Bestellungen auftreten können, ist es sinnvoll die Dauer einer etwaigen Aktivität Lieferung erst zur Laufzeit festzulegen bzw. entsprechend zu aktualisieren.

### 4.1.3 Zeitkanten

Um die zeitlichen Bedingungen eines Prozesses modellieren zu können, ist ein weiterer *Kantentyp* nötig, sogenannte *Zeitkanten*. Zwar erlauben es bereits die Kontroll- und Sync-Kanten, eine gewisse Zeitkausalität auszudrücken, doch reicht die von diesen beschriebene reine Vorgänger-Nachfolger-Beziehung für die Modellierung komplexerer Zeitbedingungen bei Weitem nicht aus.

Um die Zeitkanten einzuführen, muss die Menge der Kantentypen **eT** aus Abschnitt 2.1.3.3 um einen weiteren Typ TIME_EDGE für Zeitkanten erweitert werden. Hiermit lässt sich dann die Menge der Zeitkanten definieren als

$$Time\_e = \{e \in \mathbf{E} | \mathtt{eT}_e = \text{TIME\_EDGE}\}.$$

Weiter muss das eine Kante repräsentierende Tripel $(\mathtt{ID}_e^{start}, \mathtt{ID}_e^{dest}, \mathtt{eT}_e)$ um zwei Referenzpunkte $\mathtt{RP}_e^{start}, \mathtt{RP}_e^{dest} \in \{start, end\}$ und einen Zeitparameter $\mathtt{Time}_e$ zum 6-Tupel

$$(\mathtt{ID}_e^{start}, \mathtt{ID}_e^{dest}, \mathtt{eT}_e, \mathtt{RP}_e^{start}, \mathtt{RP}_e^{dest}, \mathtt{Time}_e)$$

erweitert werden. Die beiden Referenz-Punkte $\texttt{RP}_e^{start}$ und $\texttt{RP}_e^{dest}$ geben an, ob sich die Zeitkante auf das Start- oder End-Ereignis der Start- respektive Zielaktivität bezieht. Der Zeitparameter

$$\texttt{Time}_e =< [\texttt{Za}_e^{min}, \texttt{Za}_e^{max}], \texttt{P}_e >$$

besteht aus einem Intervall $[\texttt{Za}_e^{min}, \texttt{Za}_e^{max}] \subseteq \mathbf{TD}$ und einer Funktion für dynamische Abstände $\texttt{P}_e$. Das Intervall spezifiziert die minimale und maximale Zeitdauer, die zwischen den beiden, durch die Referenzpunkte angegebenen, Ereignissen vergehen darf. Ist $\texttt{P}_e \neq \varnothing$, so kann die Dauer der Aktivität zur Laufzeit durch die von $\texttt{P}_e$ spezifizierte Funktion eingeschränkt werden (siehe Dynamische Zeitkanten).

Die zuvor bereits definierten Kantentypen (d. h. Kontroll-, Sync- und Schleifenrücksprungkanten) erhalten für diese neuen Eigenschaften die Standardwerte $\texttt{RP}_e^{start} = end$, $\texttt{RP}_e^{dest} = start$ und $\texttt{Time}_\texttt{e} =< [0, \infty], \varnothing >$. Dabei sind die Werte der beiden Referenz-Punkte $\texttt{RP}_e^{start}$ und $\texttt{RP}_e^{dest}$ unveränderlich. Das Intervall $[\texttt{Za}_e^{min}, \texttt{Za}_e^{max}]$ und $\texttt{P}_e$ hingegen können den zeitlichen Bedingungen entsprechend angepasst werden. In diesem Fall erhält die Kante zusätzlich die Semantik einer Zeitkante. Dies wird im Abschnitt über die Kombination von Kanten genauer erläutert.

Zeitkanten werden in der graphischen Modellierung, wie die restlichen Kanten auch, durch eine gerichtete Kante dargestellt. Um diese dennoch von den anderen Kanten unterscheiden zu können, erhalten sie zusätzlich ein Uhr-Symbol (siehe Abbildung 4.2). Darüber hinaus kennzeichnet die Seite des Knotens, an der die Zeitkante beginnt bzw. endet, den entsprechenden Referenzpunkt der Kante. Beginnt die Kante also beispielsweise auf der rechten Seite des einen Knotens und endet auf der linken Seite des anderen, so gilt $\texttt{RP}_e^{start} = end$ und $\texttt{RP}_e^{dest} = start$.

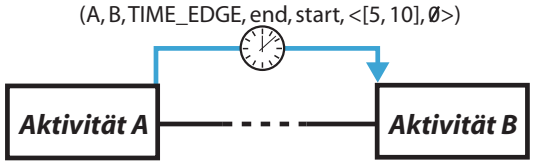



Abbildung 4.2: Darstellung einer Zeitkante

Die Verwendung einer Zeitkante ist nicht an allen Stellen und in jede Richtung innerhalb des Prozessschemas erlaubt. Beispielsweise sind Zeitkanten in der Regel nicht in entgegengesetzter Richtung des Kontrollflusses möglich. Die genauen Regeln für die einzelnen Konstrukte werden nachfolgend noch erläutert. Außerdem gibt es einige Vereinfachungsregeln, die eine kompaktere Darstellung eines, um Zeitbedingungen erweiterten, Prozessschemas erlauben.

#### 4.1.3.1 Dynamische Zeitkanten

Bei dynamischen Zeitkanten (d. h. $\mathtt{P}_e \neq \varnothing$) kann der durch die Kante repräsentierte minimale und maximale Abstand zur Laufzeit eingeschränkt werden. Dazu wird zur Laufzeit die durch $\mathtt{P}_e$ spezifizierte Funktion

$$g : [\mathbf{TD}] \rightarrow [\mathbf{TD}]$$

angewendet, welche den neuen Zeitparameter der Kante bestimmt. Die Funktion darf dabei jedoch lediglich, analog zu den dynamischen Dauern (siehe Abschnitt 4.1.2.2), den aktuellen minimalen und maximalen Abstand weiter einschränken, eine Erweiterung ist nicht möglich. Um den neuen Zeitabstand zu bestimmen, kann die Funktion beispielsweise auf ein von einer vorhergehenden Aktivität geschriebenes Datenelement oder eine externe Komponente zurückgreifen.

#### 4.1.3.2 Eindeutigkeit der Ablauffolge

Ein ausschlaggebender Punkt für die Entscheidung ob bzw. wann eine Zeitkante gezogen werden darf ist die Eindeutigkeit der Ablauffolge. Darunter versteht man, dass eine Zeitkante den durch die Kontrollkanten vorgegebenen Vorgänger-Nachfolger-Beziehungen nicht widersprechen darf. Das heißt, eine Zeitkante darf nur dann zwischen den Aktivitäten $A_i$ und $A_j$ existieren, wenn es über Kontroll- und/oder Sync-Kanten keinen vorwärts gerichteten Weg von $A_j$ nach $A_i$ gibt. Formal ausgedrückt bedeutet dies:

$$\forall e \in Time\_e : \mathtt{ID}_e^{dest} \not\leftarrow_{Control\_e \cup Sync\_e} \mathtt{ID}_e^{start}.$$

Ein Beispiel für eine Zeitkante, die dieser Regel widerspricht, ist in Abbildung 4.3 zu finden. Ausnahmen sind nur innerhalb von Schleifen möglich, da hier eine Aktivität sowohl Vorgänger als auch Nachfolger von sich selbst sein kann. Die dabei geltenden Regeln werden im Abschnitt über Schleifen erläutert (siehe Abschnitt 4.1.3.6).

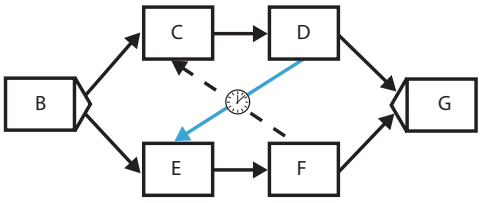



Abbildung 4.3: Eindeutigkeit der Ablauffolge

#### 4.1.3.3 Kombination von Kanten

Existiert zwischen zwei Aktivitäten mehr als eine Zeitkante, so können die auf dieselben Referenzpunkte verweisenden zusammengefasst werden. Die Semantik ist dann jene, dass aus den Zeitkanten eine neue Zeitkante hervorgeht, die der Konjunktion der einzelnen Zeitkanten entspricht. Somit gilt für zwei Zeitkanten $e_i$ und $e_j$

$$\begin{aligned} e_k &= e_i \wedge e_j \\ &= (\mathtt{ID}_k^{start}, \mathtt{ID}_k^{dest}, \mathtt{eT}_k, \mathtt{RP}_k^{start}, \mathtt{RP}_k^{dest}, < [\mathtt{Za}_i^{min}, \mathtt{Za}_i^{max}] \cap [\mathtt{Za}_j^{min}, \mathtt{Za}_j^{max}], \mathtt{P}_i \odot \mathtt{P}_j >) \end{aligned}$$

wobei $\mathtt{ID}_k^{start} = \mathtt{ID}_i^{start} = \mathtt{ID}_j^{start}$, $\mathtt{ID}_k^{dest} = \mathtt{ID}_i^{dest} = \mathtt{ID}_j^{dest}$, $\mathtt{eT}_k = \mathtt{eT}_i = \mathtt{eT}_j =$ TIME_EDGE, $\mathtt{RP}_k^{start} = \mathtt{RP}_i^{start} = \mathtt{RP}_j^{start}$ und $\mathtt{RP}_k^{dest} = \mathtt{RP}_i^{dest} = \mathtt{RP}_j^{dest}$. Die Operation $\mathtt{P}_i \odot \mathtt{P}_j$ ist dabei wie folgt definiert:

$$\mathtt{P}_i \odot \mathtt{P}_j = \begin{cases} \mathtt{P}_i & \text{falls } \mathtt{P}_j = \varnothing \\ \mathtt{P}_j & \text{falls } \mathtt{P}_i = \varnothing \\ \mathtt{P}_j \cap \mathtt{P}_i & \text{sonst} \end{cases}$$

An dieser Stelle kann es vorkommen, dass das Zeitintervall der neuen Kante leer ist. In diesem Fall sind die Zeitbedingungen des Prozessschemas inkonsistent, was wiederum zu einer Fehlermeldung bei der Überprüfung der Konsistenz durch das Zeitmanagement-System führt. Sollte dieser Fall eintreten, so müssen die Zeitkanten entsprechend angepasst werden, damit keine Inkonsistenz mehr vorhanden ist.

Haben Zeitkanten unterschiedliche Referenz-Punkte, so bleiben sie als einzelne Zeitkanten erhalten. Das bedeutet, dass zwischen zwei Aktivitäten bis zu vier verschiedene Zeitkanten existieren können. In diesem Fall ist jedoch Vorsicht bei der Festlegung der einzelnen Werte der Zeitkanten geboten, da sich die Werte der Kanten widersprechen können. Sollte dieser Umstand eintreten, wird man bei der Konsistenzüberprüfung durch eine Fehlermeldung auf diesen aufmerksam gemacht und muss dieses Problem beheben.

Existiert zwischen zwei Aktivitäten neben einer Zeitkante $e_i$ mit $\mathtt{RP}_i^{start} = end$ und $\mathtt{RP}_i^{dest} = start$ noch eine weitere Kante $e_j$ eines anderen Typs, so können diese beiden Kanten ebenfalls zusammengefasst werden, wobei die neue Kante vom Typ $\mathtt{eT}_j$ der zweiten Kante ist. Das heißt,

$$\begin{aligned} e_k &= e_i \wedge e_j \\ &= (\mathtt{ID}_k^{start}, \mathtt{ID}_k^{dest}, \mathtt{eT}_k, end, start, < [\mathtt{Za}_i^{min}, \mathtt{Za}_i^{max}] \cap [\mathtt{Za}_j^{min}, \mathtt{Za}_j^{max}], \mathtt{P}_i \odot \mathtt{P}_j >) \end{aligned}$$

wobei $\mathtt{ID}_k^{start} = \mathtt{ID}_i^{start} = \mathtt{ID}_j^{start}$, $\mathtt{ID}_k^{dest} = \mathtt{ID}_i^{dest} = \mathtt{ID}_j^{dest}$, $\mathtt{eT}_k = \mathtt{eT}_j$, $\mathtt{eT}_i =$ TIME_EDGE, $\mathtt{RP}_i^{start} = \mathtt{RP}_j^{start} = end$ und $\mathtt{RP}_i^{dest} = \mathtt{RP}_j^{dest} = start$. Diese Kombinations-Regel folgt aus dem weiter oben erwähnten Umstand, dass die anderen Kantentypen aufgrund der Art ihrer Definition um die Semantik einer Zeitkante erweitert werden können. Dabei existieren aus Sicht des Prozess-Management-Systems weiterhin zwei verschiedene Kanten zwischen den beiden Aktivitäten, die jedoch für eine bessere Übersichtlichkeit zusammengefasst wurden.

#### 4.1.3.4 Parallel-Verzweigungen

Die einzelnen Verzweigungspfade einer *Parallel-Verzweigung* unterscheiden sich nicht vom übrigen Prozessgraphen und bedürfen daher keiner speziellen Behandlung. Darüber hinaus ist es möglich Zeitkanten, zwischen verschiedenen Verzweigungspfaden anzugeben, da bei einer Parallel-Verzweigung (vgl. Abschnitt 2.1.2.4) alle Verzweigungspfade parallel durchlaufen werden und somit auch Zeitbeziehungen zwischen den Zweigen modellier- und analysierbar sein müssen. An dieser Stelle muss der Modellierer lediglich darauf achten, das Zeit- und Sync-Kanten den Bedingungen aus den vorherigen Abschnitten genügen. Verschiedene mögliche Zeitkanten in Kombination mit einer Parallel-Verzweigung zeigt Abbildung 4.4.

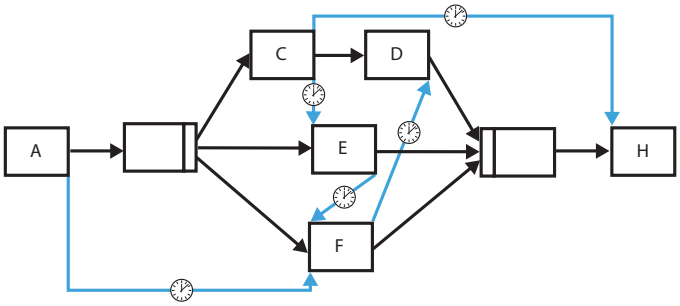



Abbildung 4.4: Zeitkanten und Parallel-Verzweigungen

#### 4.1.3.5 Alternativ-Verzweigungen

Die einzelnen Verzweigungspfade einer *Alternativ-Verzweigung* bedürfen, wie bei der Parallel-Verzweigung auch, keiner speziellen Behandlung. Es ist jedoch nicht möglich, Zeitkanten zwischen den einzelnen Pfaden einer Alternativ-Verzweigungen anzugeben, da bei jeder Ausführung des Prozessschemas höchstens einer der Pfade durchlaufen wird und hier eine zeitliche Beziehung somit semantisch nicht ohne Weiteres zu definieren ist.

In diesem Zusammenhang gibt es bei der zeitlichen Überprüfung von Alternativ-Verzweigungen aufgrund der dadurch vorhandenen verschiedenen Instanztypen einige Hindernisse, welche in Abschnitt 4.1.6 genauer beleuchtet werden.

#### 4.1.3.6 Schleifen

*Schleifen* stellen für die Definition der Semantik von Zeitkanten eine Problematik dar. Es gilt zu klären, auf welche Iteration der Schleife sich Start und Ziel einer Zeitkante beziehen. Um darüber hinaus Zeitaussagen über die nachfolgenden Aktivitäten von Schleifen treffen zu können, muss ein Weg gefunden werden, wie die maximale Dauer der Schleife ermittelt werden kann.

#### Eingehende Zeitkanten

Eingehende Zeitkanten, das heißt Zeitkanten, die an einer Aktivität $A_i$ außerhalb der Schleife starten und zu einer Aktivität $A_j$ innerhalb der Schleife führen, beziehen sich immer auf die erste Durchführung von $A_j$. Die Modellierung einer eingehenden Zeitkante, die sich auf einen der weiteren Durchläufe von $A_j$ bezieht, ist nicht möglich.

#### Ausgehende Zeitkanten

Ausgehende Zeitkanten, d. h. Zeitkanten, die an einer Aktivität $A_j$ innerhalb der Schleife beginnen und zu einer Aktivität $A_k$ außerhalb der Schleife führen, beziehen sich immer auf die letzte Durchführung von $A_j$. Auch hier ist die Modellierung einer ausgehenden Zeitkante, die sich auf einen der früheren Durchläufe von $A_j$ bezieht, nicht möglich.

#### Zeitkanten innerhalb von Schleifen

Zwischen den Aktivitäten innerhalb einer Schleife sind ebenfalls Zeitkanten modellierbar. Dabei gilt es, zwei Fälle zu unterscheiden:(1) Zeitkanten zwischen Aktivitäten, die sich in derselben Iteration befinden (2) Zeitkanten zwischen Aktivitäten, die sich in der i-ten und der i+1-ten Iteration befinden.

Für Zeitbeziehungen innerhalb derselben Iteration einer Schleife gilt, wie für Zeitbeziehungen außerhalb von Schleifen, das Kriterium der Eindeutigkeit der Ablauffolge (siehe Abschnitt 4.1.3.2). Dies bedeutet, dass eine Zeitkante hier nur in Richtung der Ablauffolge existieren kann. Abbildung 4.5 zeigt mögliche Zeitkanten innerhalb einer Iteration einer einfachen Schleife.

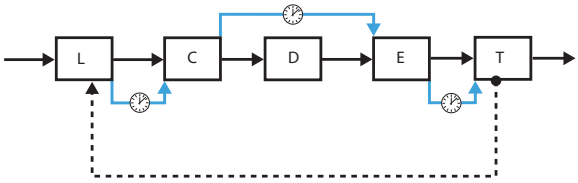



Abbildung 4.5: Mögliche Zeitkanten innerhalb einer Iteration

Für eine Zeitkante zwischen aufeinanderfolgenden Iterationen muss gelten, dass der Zielknoten Vorgänger des Quellknotens oder gleich dem Quellknoten ist. Dies hat zur Folge, dass die Zeitkante dem Kriterium der Eindeutigkeit der Ablauffolge widerspricht, wodurch sie sich eindeutig von einer Zeitkante innerhalb derselben Iteration unterscheidet. Formal ausgedrückt bedeutet dies, dass für eine Zeitkante zwischen der Aktivität $\mathtt{ID}_e^{start}$ in Iteration $i$ und $\mathtt{ID}_e^{dest}$ in Iteration $i+1$ gelten muss

$$\mathtt{ID}_e^{dest} \preccurlyeq_{Control\_e \cup Sync\_e} \mathtt{ID}_e^{start} \vee \mathtt{ID}_e^{dest} = \mathtt{ID}_e^{start}.$$

Einige mögliche Zeitkanten zwischen aufeinanderfolgenden Iterationen einer einfachen Schleife sind in Abbildung 4.6 zu sehen. Dabei ist vor allem die Zeitkante ③ interessant. Sie begrenzt den zeitlichen Abstand zwischen dem Ende der i-ten und dem Beginn der i+1-ten Iteration. ① begrenzt den Abstand zwischen dem Start der Aktivität E im i-ten und dem Ende der Aktivität C im i+1-ten Durchlauf. ② ist lediglich ein Spezialfall hiervon. Hier wird die Zeitdauer zwischen dem Start der Aktivität D in der i-ten und ihrem Start in der i+1-ten Iteration begrenzt.

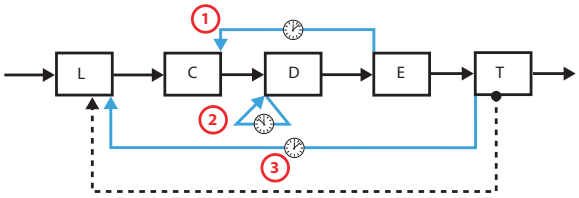



Abbildung 4.6: Mögliche Zeitkanten zwischen Iterationen

Zeitbeziehungen von der i-ten zur i+2-ten oder einer späteren Iteration sind nicht modellierbar. Zum einen gibt es keine schlüssige Möglichkeit diese graphisch in das Prozessschema einzubauen und zum anderen besteht das Problem, dass nicht entscheiden werden kann, ob die Zeitkante gültig ist oder die Schleife vor Erreichen der entsprechenden Iteration verlassen wird. Dies wiederum erschwert eine vernünftige Behandlung von Zeitfehlern sehr.

#### Dauer eines Schleifendurchlaufs und Gesamtdauer einer Schleife

Eine Zeitkante vom Ende des STARTLOOP-Knotens zum Anfang des ENDLOOP-Knotens (Ende-Start-Beziehung) begrenzt die Dauer *einer* Schleifen-Iteration (siehe ② in Abbildung 4.7).

Will man dagegen die *Gesamtdauer* der Schleife, das heißt die Summe der Dauern aller Iterationen, begrenzen, so muss eine Zeitkante zwischen dem Anfang des STARTLOOP-Knotens und dem Ende des ENDLOOP-Knotens gezogen werden (Start-Ende-Beziehung) (siehe ① in Abbildung 4.7).

Die Semantik dabei ist, dass sich der Start des STARTLOOP-Knotens sowie das Ende des ENDLOOP-Knotens außerhalb des Schleifen-Rumpfes befinden, wohingegen sich das Ende des STARTLOOP-Knotens und der Anfang des ENDLOOP-Knotens innerhalb der Schleife befinden und bei jeder Iteration durchlaufen werden.

Als Alternative zu dieser Darstellung ist das Einfügen zusätzlichen Nullknoten, d. h. Knoten ohne Aktivität der Dauer 0, möglich. Zur Begrenzung der Gesamtdauer werden diese vor bzw. hinter dem STARTLOOP- bzw. ENDLOOP-Knoten eingefügt und zur Begrenzung der Dauer einer Iteration hinter bzw. vor dem STARTLOOP- bzw.

ENDLOOP-Knoten und jeweils über eine Zeitkante der Dauer 0 mit dem entsprechenden Knoten der Schleife verbunden. Allerdings leidet die Übersichtlichkeit des Prozessgraphen unter diesem Vorgehen.

Eine weitere Möglichkeit zur Begrenzung der Gesamtdauer einer Schleife ist die Angabe der Dauer für eine Iteration zusammen mit der minimalen und maximalen Anzahl an Durchläufen der Schleife (beispielsweise in den Meta-Daten des ENDLOOP-Knotens). Hieraus kann dann die Gesamtdauer der Schleife berechnet werden. Da dies jedoch äquivalent zur Angabe einer Gesamtdauer ist, wird diese Möglichkeit nicht weiter diskutiert.

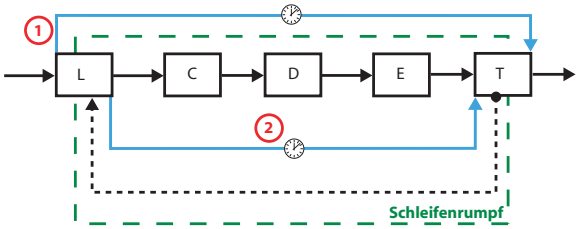



Abbildung 4.7: Dauer eines Schleifendurchlaufs und Gesamtdauer einer Schleife

### 4.1.4 Zeitzustände

Sobald ein Prozess zur Ausführung gebracht wird, das heißt aus dem Prozessschema eine Prozessinstanz erzeugt wird, ist es nötig, den Zeitzustand der Aktivitäten beziehungsweise Knoten zu überwachen. Hierzu muss innerhalb der Knoten-Definitionen der Prozessinstanzen (siehe Abschnitt 2.1.3.2 und 2.1.4) die Definition des Zeit-Elements

$$\texttt{Time}_n = ([\texttt{FZ}_n^{begin}, \texttt{SZ}_n^{begin}], [\texttt{FZ}_n^{end}, \texttt{SZ}_n^{end}], [\texttt{D}_n^{min}, \texttt{D}_n^{max}], \texttt{K}_n, \texttt{P}_n)$$

um ein Element $\texttt{Ts}_n \in \mathbf{TS}$ für den Zeitzustand erweitert werden. Dieser Zeitzustand ist weitgehend orthogonal zum Knotenzustand aus Abschnitt 2.1.4, jedoch sind nicht alle Zustands-Kombinationen sinnvoll. Ein Zustandsübergangsdiagramm der Zeitzustände in Kombination mit den Aktivitätszuständen ist in Abbildung 4.8 zu finden. Die möglichen Zeitzustände sind im Einzelnen:

**UNDEFINED** ist der Standardzustand der Knoten und bedeutet, dass für diesen Knoten momentan keine Aussage bezüglich der Einhaltung seines Termins getroffen werden kann. Er ist nur mit den Knotenzuständen NOT_ACTIVATED, FAILED und SKIPPED sinnvoll kombinierbar.

**ON_TIME** bedeutet für die Knotenzustände STARTED und SUSPENDED, das der aktuelle Zeitpunkt im Ausführungsfenster des Knotens liegt. Für den Knotenzustand COMPLETED bedeutet er, dass der Knoten pünktlich beendet wurde und

für ACTIVATED bedeutet er, dass die aktuelle Uhrzeit im Startzeitfenster des Knotens liegt, dieser jetzt also problemlos gestartet werden könnte. Mit anderen Knotenzuständen ist ON_TIME nicht sinnvoll kombinierbar.

**EARLY** hat in Zusammenhang mit dem Knotenzustand ACTIVATED die Bedeutung, dass die Aktivität zwar aufgrund des Kontrollflusses gestartet werden könnte, die aktuelle Uhrzeit aber noch vor dem berechneten Startzeitfenster der Aktivität liegt und es somit eventuell zu zeitlichen Problemen kommen könnte. Dies ähnelt der Bedeutung im Zusammenspiel mit den Knotenzuständen STARTED und SUSPENDED, nämlich dass die Aktivität bereits gestartet wurde, der aktuelle Zeitpunkt aber noch vor ihrem Endzeitfenster liegt, das heißt die minimale Dauer der Aktivität noch nicht vergangen ist. Wird ein Knoten vor Erreichen seines Endzeitfensters beendet, so erhält er im Knotenzustand COMPLETED ebenfalls den Zeitzustand EARLY. Auch hier sind mit den anderen Knotenzuständen keinen sinnvollen Kombinationen möglich.

**LATE** besagt in Verbindung mit dem Knotenzustand NOT_ACTIVATED, dass der Knoten noch nicht ausgeführt werden kann, sein Startzeitfenster aber bereits überschritten ist. In Verbindung mit ACTIVATED besagt er, dass die Aktivität noch nicht gestartet wurde, dies aber bereits hätte geschehen müssen. Bei den Knotenzuständen RUNNING und SUSPENDED ist die Bedeutung die, dass das Endzeitfenster bereits überschritten wurde, die Aktivität aber noch nicht abgeschlossen ist. Ähnlich ist die Aussage im Zusammenhang mit COMPLETED, dass der Knoten erst nach Ablauf des Endzeitfensters beendet wurde. Mit den Knotenzuständen FAILED und SKIPPED ist keine Kombination möglich.

Die Einführung der Zeitzustände zusätzlich zu den Ausführungszuständen ist nötig, da die Ausführungszustände nicht ausreichen, um den Zeitstatus der Aktivitäten wiederzugeben. Ein Beispiel hierfür ist die Ausführungszustand-Zeitzustand-Kombination ACTIVATED/EARLY. Semantisch entspricht dieser Zustand dem Ausführungszustand NOT_ACTIVATED, da nicht alle Eingangsbedingungen des Knotens erfüllt sind und dieser daher noch nicht gestartet werden kann. Andererseits sind die Zeitbedingungen eines Prozesses lediglich Vorgaben, die auf expliziten Wunsch des Benutzers ignoriert werden können. Dies sollte jedoch nur in Ausnahmesituationen geschehen, da in diesem Fall ein Zeitfehler ausgelöst werden muss. Daher ist der Zustand ACTIVATED ebenfalls ungeeignet für die Wiedergabe dieser Situation. Es ist somit nötig eine zusätzliche Unterteilung der Ausführungszustände einzuführen, wie dies mithilfe der Zeitzustände geschieht.

### 4.1.5 Kontroll- vs. Zeitkanten

Kontroll- und Sync-Kanten unterscheiden sich auf der syntaktischen Ebene und in ihrer Semantik für die Modellierung kaum von Zeitkanten. Hierdurch sind für die Modellierung

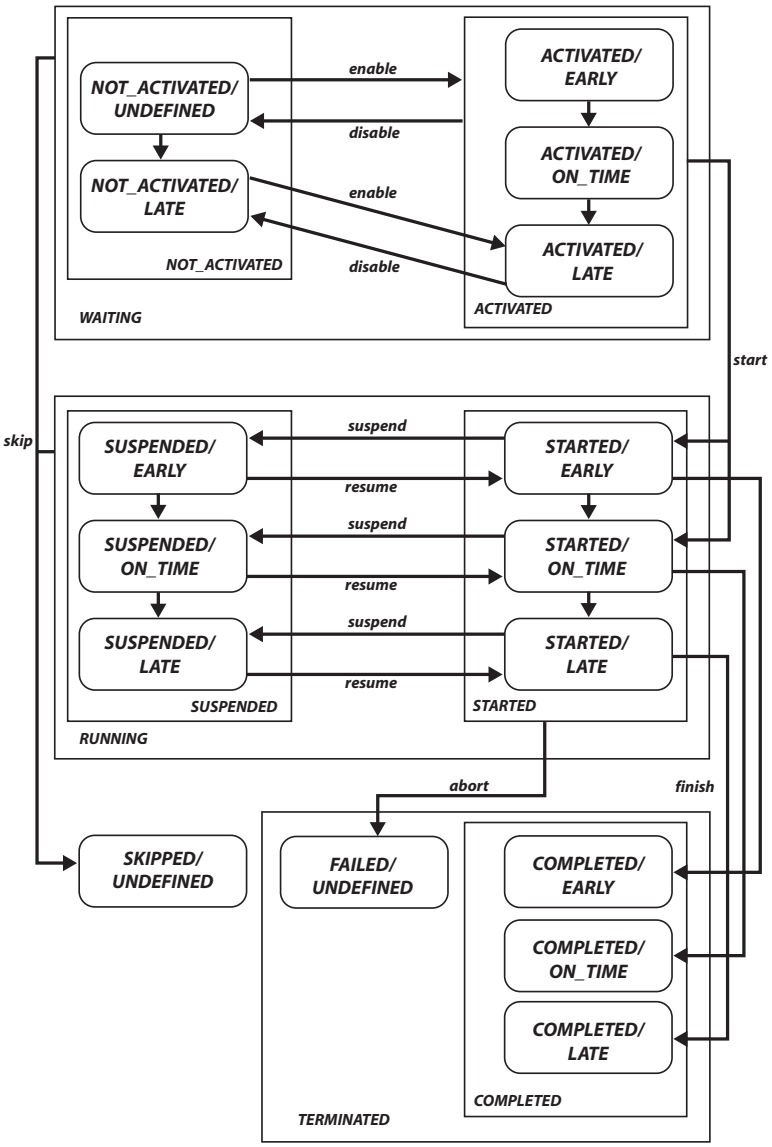
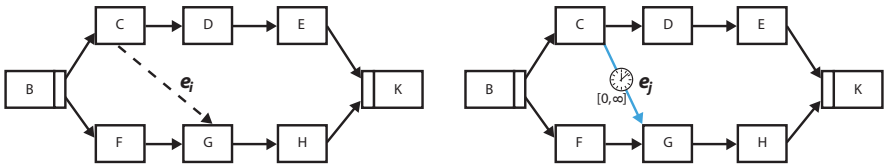



Abbildung 4.8: Zustandsübergangsdiagramm der Zeitzustände

des Kontrollflusses und der Zeitbedingungen Kontroll- und Sync-Kanten weitestgehend durch Zeitkanten ersetzbar. Jedoch unterscheidet sich ihre Semantik zur Laufzeit, wie Beispiel 4.1 verdeutlicht, weshalb eine Trennung von Kontroll-/Sync-Kanten und Zeitkanten nötig ist.

Die Abbildung zeigt zwei Ausschnitte eines Prozesses. Dabei wurde bei dem auf der rechten Seite eine Sync-Kante durch eine entsprechende Zeitkante ersetzt.




Von beiden Kanten wird beschrieben, dass die Aktivität G erst gestartet werden kann, sobald die Aktivität C beendet wurde. Auch die Definition der Sync-Kante

$$e_i = (C, G, \text{SYNC\_EDGE}, end, start, < [0, \infty], \varnothing >)$$

entspricht, bis auf den Kantentyp, der Definition der Zeitkante

$$e_j = (C, G, \text{TIME\_EDGE}, end, start, < [0, \infty], \varnothing >).$$

Erst bei der Betrachtung der Zustände der Knoten kann man einen Unterschied erkennen. Für den Fall der Sync-Kante gilt nach dem Beenden des Knoten F bei zeitgleicher Ausführung von C Folgendes:

$$\begin{aligned} \texttt{Ns}_B = \text{COMPLETED}, \texttt{Ns}_F = \text{COMPLETED} \text{ und } \texttt{Ns}_C = \text{RUNNING} \\ \Rightarrow \texttt{Ns}_G = \text{NOT\_ACTIVATED} \text{ und } \texttt{Ts}_G = \text{UNDEFINED}. \end{aligned}$$

Für die Zeitkante gilt hingegen:

$$\begin{aligned} \texttt{Ns}_B = \text{COMPLETED}, \texttt{Ns}_F = \text{COMPLETED} \text{ und } \texttt{Ns}_C = \text{RUNNING} \\ \Rightarrow \texttt{Ns}_G = \text{ACTIVATED} \text{ und } \texttt{Ts}_G = \text{EARLY} \end{aligned}.$$

Das heißt, im Fall der Zeitkante würde die Aktivität G bereits in der Arbeitsliste angeboten werden, obwohl die Aktivität C noch nicht beendet wurde. Sie könnte somit trotz eventueller Warnungen des Systems gestartet werden. Dies ist im Fall der Sync-Kante nicht möglich, da hier die Aktivität G erst nach Beenden der Aktivität C in der Arbeitsliste zur Ausführung erscheint.

Beispiel 4.1: Unterschied Kontroll-/Zeitkanten

Die in Abschnitt 4.1.3.3 vorgestellte Kombination von Kontroll- und Sync-Kanten mit Zeitkanten dient einer einfacheren Modellierbarkeit, aber von der Ausführungssemantik aus betrachtet entspricht dieses Kombi-Konstrukt immer noch zwei getrennten Kanten, die lediglich durch ein gemeinsames Symbol repräsentiert werden.

### 4.1.6 Behandlung verschiedener Instanztypen

Im Zusammenhang mit Alternativ-Verzweigungen und Schleifen besteht ein Problem darin, dass Instanzen mit verschiedenen Eigenschaften, man spricht hierbei von *Instanztypen* (siehe Abschnitt 2.1.1), in einem Prozessschema vereint werden (siehe Abbildung 4.9). Dies stellt das Zeitmanagement oder genauer gesagt die Berechnung der Zeiteigenschaften des Prozessschemas vor eine große Herausforderung, da die einzelnen Instanztypen unterschiedliche Zeitbedingungen aufweisen können. In den gemeinsamen Bereichen nach den Alternativ-Verzweigungen und Schleifen überlagern sich die daraus resultierenden verschiedenen Zeitwerte jedoch, sodass den nachgelagerten Aktivitäten keine eindeutigen Zeitpunkte mehr zugewiesen werden können. Eine einfache Illustration dieser Problematik ist in Beispiel 4.2 zu finden.

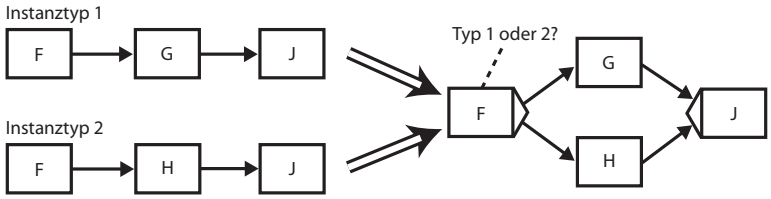
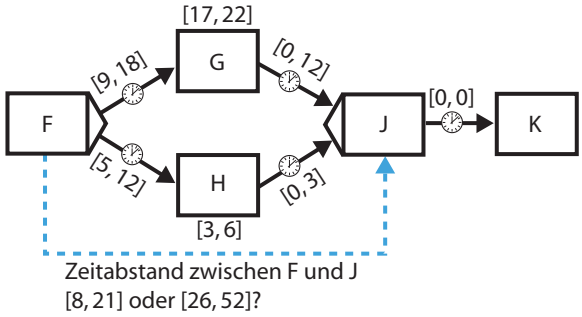



Abbildung 4.9: Vereinigung unterschiedlicher Instanztypen

Dieser Umstand hat zur Folge, dass manche Zeitkanten eventuell nicht mehr vollständig auf ihre Korrektheit überprüft werden können. Dieses *Prüfbarkeitsproblem* ist eine der größten Hürden für das Zeitmanagement bei Prozess-Management-Systemen, da daraus resultierend die Korrektheit eines Prozessschemas möglicherweise nicht mehr garantiert werden kann. In der Literatur zu Zeitmanagement-Systemen gibt es verschiedene Ansätze, um mit diesen Problemen umzugehen (siehe auch Kapitel 3):

**kürzester und längster Pfad / optimistischer Ansatz** Sowohl Marjanovic und Orlowska [Marjanovic & Orlowska, 1999] als auch Pozewaunig [Pozewaunig, 1996] schlagen vor, lediglich den *kürzesten und den längsten Pfad* durch das Prozessschema zu betrachten. Die daraus resultierenden Werte werden dann jeweils als frühester oder spätester Zeitpunkt für die Aktivitäten verwendet. Einen ähnlichen Ansatz verfolgt Grimm [Grimm, 1997] unter der Bezeichnung *optimistischer Ansatz*. Dabei werden, bei der Berechnung der Zeitpunkte der Aktivitäten, für die XOR-Join-Knoten die

Im unten dargestellten Prozessschema weisen die beiden Pfade der Alternativ-Verzweigung signifikant unterschiedliche Dauern auf. Der obere Pfad benötigt zwischen 26 und 52 Zeiteinheiten und der untere zwischen 8 und 21.

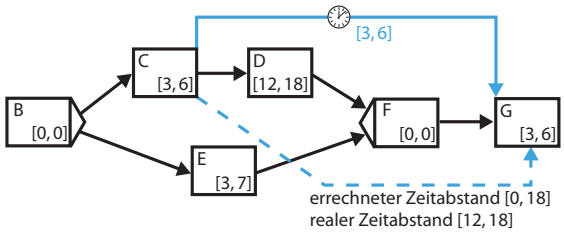



Es stellt sich die Frage, welchen Abstand zum Knoten $F$ beziehungsweise welchen Zeitpunkt man dem Knoten $J$ und dessen Nachfolger zuweist. Weist man $J$ eines der beiden Intervalle der alternativen Pfade zu, kommt es zu einem Zeitfehler, sobald der andere Pfad ausgewählt wird. Weist man $J$ jedoch das Intervall $[8, 52]$ zu, so scheinen nach den Zeitangaben für $J$ auch die Zeitpunkte $[22, 25]$ zulässig zu sein, welche aber in keinem der beiden Pfade möglich sind.

Beispiel 4.2: Prüfbarkeitsproblem

Werte mit den „schwächsten“ zeitlichen Bedingungen verwendet. Das bedeutet, wenn $\texttt{Za}_i^{min}$ und $\texttt{Za}_i^{max}$ die Werte des minimalen und maximalen Zeitabstandes des Pfades $i$ zwischen den beiden Split-/Join-Knoten sind, so gilt für den Zeitabstand zwischen den beiden Block-Knoten $\texttt{Za}_{neu}^{min} = min_i(\texttt{Za}_i^{min})$ und $\texttt{Za}_{neu}^{max} = max_i(\texttt{Za}_i^{max})$. Ein Problem dieser beiden Ansätze ist, wie man an Beispiel 4.3 sehen kann, dass nicht alle Inkonsistenzen der Zeitmodellierung aufgedeckt werden können.

**pessimistischer Ansatz** Grimm [Grimm, 1997] schlägt als Alternative zum optimistischen Ansatz noch eine weitere Variante vor, die er als *pessimistischen Ansatz* bezeichnet. Dabei werden bei der Berechnung der Zeitpunkt der Aktivitäten für die XOR-Join-Knoten die Werte mit den „stärksten“ zeitlichen Bedingungen verwendet. Das heißt die XOR-Join-Knoten erhalten $\texttt{Za}_{neu}^{min} = max_i(\texttt{Za}_i^{min})$ und $\texttt{Za}_{neu}^{max} = min_i(\texttt{Za}_i^{max})$ als Abstand zum entsprechenden XOR-Split-Knoten. Der Nachteil dieses Ansatzes besteht darin, dass sehr viele Prozessschemata als zeitlich inkonsistent markiert werden, obwohl diese vollkommen korrekt sind. Ein Beispiel hierfür ist das in Beispiel 4.2 gezeigte Prozessschema. Bei diesem wird als Abstand zwischen F und




Für das oben dargestellte Prozessschema berechnet der optimistische Ansatz aus den Zeitdaten der Aktivitäten B, C, F und G einen Abstand von $[0, 18]$ zwischen den Aktivitäten C und G. Dieser ist mit der Zeitkante $[3, 6]$ zwischen C und G kompatibel. Betrachtet man jedoch die Aktivität D, so erkennt man, dass der reale Abstand zwischen C und G aufgrund der Dauer der Aktivität D mindestens 12 ist und somit die Zeitkante zwischen den beiden Aktivitäten niemals eingehalten werden kann.

Beispiel 4.3: Probleme des optimistischen Ansatzes

J ein leeres Intervall $\emptyset$ ermittelt. Dies führt dazu, dass das Prozessschema als inkonsistent deklariert wird, obwohl es problemlos ausführbar ist.

**Instanztypen einzeln betrachten** Bei Bettini et al. [Bettini *et al.*, 2002b] findet sich ein Ansatz, bei dem alle Instanztypen einzeln betrachtet werden. Dazu zerlegt Bettini ein Prozessschema an dessen XOR-Splits[1] wieder in seine einzelnen Instanztypen. Diese werden dann nacheinander auf zeitliche Inkonsistenz überprüft. Dieser Ansatz hat gleich mehrere Nachteile. Zum einen steigt die Anzahl der Instanztypen exponentiell mit der Anzahl der im Prozessschema vorhandenen Alternativ-Verzweigungen an, was diese Variante extrem rechenintensiv macht, zum anderen ist es nach Berechnung der Werte der einzelnen Instanztypen eventuell nicht mehr möglich, diese in einem Prozessschema zu kombinieren, da sich durch gewisse Zeitbedingungen auch die Zeiten von Aktivitäten vor den Alternativ-Verzweigungen verändert haben.

**Unfolded Workflow Graph** Eder et al. [Eder *et al.*, 2000] nennen einen weiteren Ansatz, der dem von Bettini nicht unähnlich ist. Jedoch werden hierbei die einzelnen Instanztypen nicht vollständig getrennt, sondern es wird nur für jeden Instanztyp ein eigener Teil-Graph sowie ein eigener Endknoten eingeführt. Dazu werden die Bereiche in denen sich die Instanztypen unterscheiden dupliziert und erneut in das Prozessschema eingefügt. Ein einfaches Beispiel für den daraus resultierenden neuen Graphen ist in Abbildung 4.10 zu finden.

[1] Schleifen werden nicht betrachtet

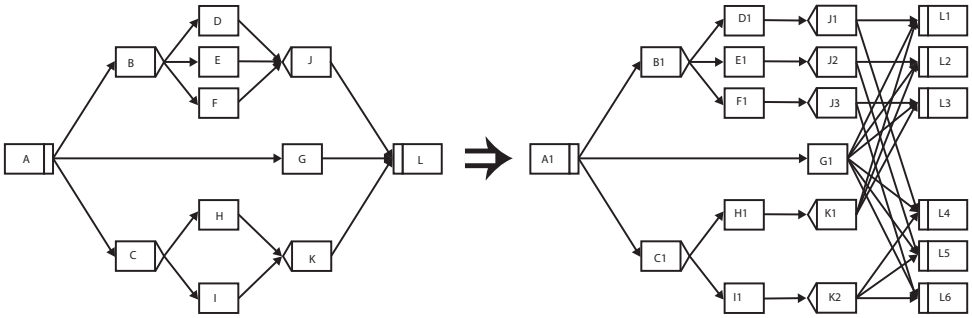



Abbildung 4.10: Unfolded Workflow Graph [Eder *et al.*, 2000]

Durch dieses Vorgehen wird vermieden, dass sich die Zeiteigenschaften der Instanztypen in den der Trennung vorgelagerten Bereichen unterscheiden. Allerdings hat auch dieser Ansatz das Problem, dass der neue Graph gegenüber dem alten exponentiell in der Anzahl der Alternativ-Verzweigungen anwächst und somit erheblich mehr Rechenaufwand benötigt.

Jede der vorgestellten Lösungen hat ihre Vor- und Nachteile, weswegen sich der Implementierer eines Zeitmanagement-Systems für eine oder mehrere der Alternativen entscheiden muss. Der Unfolded Workflow Graph von Eder et al. ist zwar vollständig und liefert korrekte Ergebnisse, jedoch stellt das exponentielle Wachstum des Graphen und damit auch der exponentielle Zeitbedarf des Berechnungsalgorithmus ein großes Problem dar. Zur Modellierungszeit mag dies möglicherweise noch tolerabel sein, jedoch wäre zur Laufzeit die Prozessorlast durch das Zeitmanagement-System viel zu hoch und dieses damit höchstwahrscheinlich zu träge. Dasselbe gilt auch für den Vorschlag von Bettini et al., der zusätzlich nicht bei jedem Prozessschema problemlos angewendet werden kann. Der pessimistische Ansatz von Grimm hat dagegen das Problem, das zu viele korrekte Prozessschemata als inkonsistent abgelehnt werden, wohingegen bei der Verwendung des optimistischen Ansatzes oder des kürzesten und längsten Pfades einige Inkonsistenzen nicht erkannt werden.

Eine Möglichkeit das gewünschte Ergebnis zu erreichen ist die Kombinationen mehrerer Alternativen: Zum Modellierungszeitpunkt wird der Unfolded Workflow Graph aus [Eder *et al.*, 2000] verwendet, da hier Schnelligkeit und Ressourcenschonung nicht von zentraler Bedeutung sind, es aber wichtig ist, Inkonsistenzen korrekt zu erkennen. Während der Laufzeit hingegen, wenn die ermittelten Daten nur noch aktualisiert werden, genügt es den optimistischen Ansatz beziehungsweise den Ansatz des kürzesten und längsten Pfades zu verwenden. Zum einen ist zu diesem Zeitpunkt aufgrund der Berechnungen zur Modellierungszeit bekannt, dass das Prozessschema zu Beginn konsistent war, und zum anderen ist hier ein sinnvoller Umgang mit den Ressourcen von hoher Priorität.

## 4.2 Zeitplanung

Es gibt verschiedene Möglichkeiten die zeitliche Ebene eines Prozesses zu repräsentieren. Einige davon stammen aus der *Netzplantechnik*, andere aus der Familie der *Temporal Constraint Networks* und wieder andere aus der Gruppe der *Petri-Netze* (siehe Abschnitt 3.1). Die im Folgenden vorgestellte Variante ist eine Erweiterung des sogenannten *Simple Temporal Problem* (*STP*) [Dechter *et al.*, 1991] aus der Gruppe der Temporal Constraint Networks. Diese wird sowohl um einige Konzepte aus der Netzplantechnik, als auch um einige für den Fall der Prozess-Modellierung nötige Konzepte erweitert.

Die *Simple Temporal Problems* wurden ausgewählt, da sie gegenüber den anderen Verfahren verschiedene Vorteile bieten. Beispielsweise verwenden *Simple Temporal Problems* Intervalle für die Angabe der Zeitbedingungen. Dies ist für eine vollständige Behandlung von Zeitfehlern von großem Vorteil, da sowohl Zeitüber- als auch -unterschreitungen behandelt werden können. Zugleich ist es möglich Zeitfenster für den Start bzw. das Ende einzelner Aktivitäten zu bestimmen, was für eine Vorhersage von Zeitdaten und die Optimierung zeitlicher Abläufe notwendig ist. Die Möglichkeit implizit vorhandene Zeitbedingungen zu ermitteln, erlaubt einen tiefer gehenden Einblick und eine genauere Analyse der zeitlichen Abhängigkeiten eines Prozesses. Weiterhin existieren für *Simple Temporal Problems* effiziente Algorithmen für die Feststellung ihrer Konsistenz.

Dieser Abschnitt gliedert sich wie folgt: Zunächst wird in Abschnitt 4.2.1 das verwendete Zeitmodell vorgestellt und definiert. Dieses basiert wie erwähnt auf STPs und ermöglicht die einfache Definition der benötigten Algorithmen.

Abschnitt 4.2.2 erläutert, wie im Rahmen des Zeitmodells mit unterschiedlichen Zeiteinheiten verfahren wird. Hierzu wird eine Transformation angegeben, mit der die unterschiedlichen Zeiteinheiten in eine gemeinsame Basis-Zeiteinheit umgerechnet werden.

Abschnitt 4.2.3 beschreibt die Transformation eines Prozessschemas und seiner Konstrukte in das zuvor definierte Zeitmodell.

Nach der Transformation wird das Zeitmodell und damit das Prozessschema auf zeitliche Konsistenz überprüft und Schedules für die Aktivitäten berechnet. Die hierfür nötigen Algorithmen werden in Abschnitt 4.2.4 vorgestellt.

Die Kombination von zwei der zuvor vorgestellten Möglichkeiten zur Behandlung unterschiedlicher Instanztypen mit dem Zeitmodell und dessen Algorithmen findet in Abschnitt 4.2.5 statt.

Schließlich wird in Abschnitt 4.2.6 eine Möglichkeit vorgestellt, um aus einem zeitlich inkonsistenten Prozessschema durch eine Lockerung der Zeitbedingungen automatisiert ein konsistentes Zeitmodell zu erhalten.

### 4.2.1 Das Zeitmodell

Das Zeitmodell wird, ebenso wie das Prozessschema, durch einen Graphen dargestellt. Die Knoten des Zeit-Graphen entsprechen den Ereignissen im Ablauf des Prozesses, das

heißt den Start- und End-Ereignissen der Prozessaktivitäten. Jedem Ereignis wird ein eindeutiger Bezeichner $E_i$ sowie ein frühester und ein spätester Zeitpunkt – $FZ_i$ bzw. $SZ_i$ – zugeordnet. Knoten des Zeit-Graphen können beliebig durch Kanten verbunden werden. Diese werden mit einem Intervall vom minimalen bis zum maximalen Abstand zwischen den beiden Ereignissen $[D_{ij}^{min}, D_{ij}^{max}]$ beschriftet (siehe Abbildung 4.11).

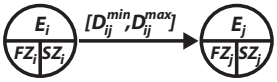



Abbildung 4.11: Knoten und Kante des Zeitmodells

Im Sinne der Temporal Constraint Networks stellt jeder Knoten $E_i$ des Zeitmodells eine Variable $X_i$ dar, die einen kontinuierlichen Wertebereich $Dom(X_i) \subseteq [-\infty, \infty]$ besitzt. Die Kanten repräsentieren Einschränkungen der möglichen Werte einer Variablen gegenüber den Werten einer anderen Variable. Sie werden deshalb als *Constraints* bezeichnet.

**Definition 4.1** (Constraint, zusammenhängendes Constraint [Dechter *et al.*, 1991])**.**

- Ein (binäres) *Constraint* ist eine Menge von diskreten Intervallen

  $$c_{ij} = \{[a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n]\} \subseteq \mathbb{N}.$$

  Es schränkt die erlaubten Werte des Abstands der beiden Variablen $X_i$ und $X_j$ ein und repräsentiert die Disjunktion

  $$c_{ij} = (a_1 \leq X_j - X_i \leq b_1) \vee (a_2 \leq X_j - X_i \leq b_2) \vee \ldots \vee (a_n \leq X_j - X_i \leq b_n).$$

  Dabei wird angenommen, dass alle Intervalle geschlossen und paarweise verschieden sind.

- Ein *zusammenhängendes Constraint* ist ein Constraint, bei dem die Menge aus genau einem Intervall besteht

  $$c_{ij} = \{[a, b]\}$$

Einfach formuliert hat ein zusammenhängendes Constraint $c_{ij}$ die Bedeutung, dass der Abstand zwischen beliebigen Werten von $X_i$ und $X_j$ mindestens $a$ und höchsten $b$ Zeiteinheiten betragen darf. Für den hier betrachteten Spezialfall der Simple Temporal Problems gilt sowohl für die Wertebereiche als auch für die Constraints, dass diese zusammenhängend sein müssen. Im Folgenden handelt sich daher bei allen Constraints immer um zusammenhängende Constraints, auch wenn dies nicht explizit angegeben ist.

Das auf den Simple Temporal Problems basierende Zeitmodell ist wie folgt definiert:

**Definition 4.2** (Zeitmodell)**.** Ein *Zeitmodell* ist ein Tripel $(\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$, wobei

- $\mathbf{V} = \{E_0, \ldots, E_n\}$ eine Menge von Knoten,
- $\mathbf{A} \subseteq \{(E_i \rightarrow E_j) | E_i, E_j \in \mathbf{V}\}$ eine Menge von gerichteten Kanten und
- $\mathbf{\Gamma} = \{c_{ij} | 0 \leq i, j \leq n\}$ eine Menge zusammenhängender Constraints ist.

$\mathbf{\Gamma}$ weist jeder Kante $(E_i \rightarrow E_j) \in \mathbf{A}$ ein Constraint $c_{ij}$ zu. Außerdem enthält $\mathbf{\Gamma}$ für jedes $(E_i \rightarrow E_j) \in \mathbf{A}$ ein implizites Constraint $c_{ji} = -c_{ij}$ in entgegengesetzter Richtung und für jedes $(E_i \rightarrow E_j) \in \{(E_i \rightarrow E_j) | E_i, E_j \in \mathbf{V}\} \setminus \mathbf{A}$ ein implizites Constraint $c_{ij} = [-\infty, \infty]$.

Der Wertebereich eines Knotens $Dom(E_i)$ des Zeitmodells wird als Constraint $c_{0i} = Dom(E_i) = [FZ_i, SZ_i]$ zwischen dem Quellknoten des Zeitmodells $E_0$ und dem jeweiligen Knoten $E_i$ aufgefasst. Dabei besitzt der Quellknoten $E_0$ den festen Wertebereich $Dom(E_i) = [0, 0]$. Im Folgenden ist daher, wenn vom Wertebereich eines Knotens die Rede ist, immer ein solches Constraint $c_{0i}$ zum Startknoten des Zeitmodells gemeint.

Im Gegensatz zu den Knoten eines Prozessschemas besitzen die Ereignisse eines Zeitmodells selbst keine Dauer. Damit unterscheidet das Zeitmodell auch nicht mehr zwischen der Dauer einer Aktivität und den Abständen zwischen Aktivitäten. Ein einfaches Zeitmodell mit entsprechendem Zeitgraphen ist in Beispiel 4.4 zu finden.

Mithilfe dieses Modells ist es möglich, die Konsistenz des Prozessschemas zu überprüfen. In einem weiteren Schritt kann dann ein sogenannter *Schedule* berechnet werden, welcher jeder Variablen einen Wert beziehungsweise Wertebereich zuweist, sodass alle Constraints erfüllt sind[2]. Die Werte des Schedules stellen eine Lösung des durch das Zeitmodell repräsentierten STPs dar und sind somit zulässige Zeitpunkte für das Eintreten der jeweiligen Ereignisse im Prozess, sodass alle Zeitbedingungen erfüllt werden.

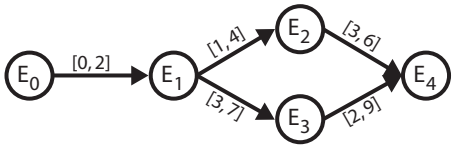
Dieses Zeitmodell reicht bereits aus um die zeitlichen Eigenschaften der einfachen Konstrukte des Prozess-Metamodells, wie Sequenz, Parallelität und beliebige Kontroll-, Sync- und Zeit-Kanten, vollständig beschreiben zu können. Für die beiden verbliebenen Konstrukte Alternativ-Verzweigung und Schleifen wird das Zeitmodell an geeigneter Stelle erweitert, bzw. es wird bei der Transformation in das Zeitmodell eine passende Repräsentation gefunden werden müssen. Ob und wie das Zeitmodell beziehungsweise die Transformation erweitert werden muss, hängt vom verwendeten Ansatz für die Behandlung von Alternativen ab (siehe Abschnitt 4.1.6). Einige der Varianten werden zusammen mit den nötigen Erweiterungen in Abschnitt 4.2.5 genauer ausgeführt. Da es hierbei jedoch auch nötig ist, die verwendeten Algorithmen zu modifizieren, werden zunächst die Basis-Algorithmen zur Transformation eines Prozessschemas in das Zeitmodell sowie zur Zeitrechnung auf diesem vorgestellt.

[2] Unter der Voraussetzung, dass diese Werte überhaupt existieren

Das Zeitmodell[a] $TM = (\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$ mit $\mathbf{V} = \{E_0, E_1, E_2, E_3, E_4\}$, $\mathbf{A} = \{(E_0 \rightarrow E_1), (E_1 \rightarrow E_2), (E_1 \rightarrow E_3), (E_2 \rightarrow E_4), (E_3 \rightarrow E_4)\}$ und

$$\mathbf{\Gamma} = (c_{ij}) = \begin{bmatrix} [0,0] & [0,2] & [-\infty,\infty] & [-\infty,\infty] & [-\infty,\infty] \\ [-2,0] & [0,0] & [1,4] & [3,7] & [-\infty,\infty] \\ [-\infty,\infty] & [-4,-1] & [0,0] & [-\infty,\infty] & [3,6] \\ [-\infty,\infty] & [-7,-3] & [-\infty,\infty] & [0,0] & [2,9] \\ [-\infty,\infty] & [-\infty,\infty] & [-6,-3] & [-9,-2] & [0,0] \end{bmatrix}$$

repräsentiert den unten dargestellten Zeitgraphen. Hierbei werden Constraints mit einem Wertebereich von $[-\infty, \infty]$ nicht dargestellt, da sie zum einen keine entsprechende Kante in $\mathbf{A}$ besitzen und zum anderen keinerlei Einschränkung darstellen.




[a] Das dargestellte Zeitmodell dient lediglich zur Verdeutlichung der Eigenschaften eines Zeitmodells und entspricht keinem Prozessschema.

Beispiel 4.4: Zeitmodell

### 4.2.2 Behandlung unterschiedlicher Zeiteinheiten

Wie in Abschnitt 3.2 diskutiert, existieren bisher keine zufriedenstellende Ansätze zur Behandlung unterschiedlichen *Zeiteinheiten*. Aus diesem Grund werden Dauern, Abstände und Fristen zwar wie in Abschnitt 4.1.1 angegeben unter Zuhilfenahme verschiedener Zeiteinheiten spezifiziert, um mit diesen zu rechnen werden sie jedoch zunächst in eine Basis-Zeiteinheit transformiert.

Als Basis-Zeiteinheit fungiert im Folgenden die Einheit Minuten, die beschriebene Umrechnung lässt sich jedoch ohne Weiteres auf andere Basis-Zeiteinheiten übertragen.

Wurde eine Zeitdauer unter Verwendung verschiedener Zeiteinheiten angegeben, so wird diese zunächst in die kleinste der verwendeten Zeiteinheiten umgerechnet. Das heißt beispielsweise, dass die Zeitdauer „1 Tag 3 Stunden“ in „27 Stunden“ umgerechnet wird. Im nächsten Schritt wird diese Zeitdauer in die Basis-Zeiteinheit umgerechnet. Dabei findet keine direkte Umrechnung statt, sondern die Dauer wird zu einem Intervall von Zeitdauern der Basis-Zeiteinheit. Hierzu wird die untere Grenze des Intervalls derart gewählt, dass sie der kleinste Wert ist, für den man bei einer erneuten Umrechnung in die ursprüngliche Zeiteinheit unter Rundung gerade noch den ursprünglichen Wert erhalten

würde. Ebenso wird die obere Grenze so gewählt, dass sie dem größten Wert entspricht, für den dies gilt. Das heißt, es gilt beispielsweise $27h = [26, 5h, 27, 49h] = [1590min, 1649min]$ und $1d = [0, 5d, 1, 49d] = \ldots = [720min, 2159min]$.

Die Umrechnung in ein Intervall findet nur für Zeitdauern statt. Bei Zeitpunkten erhält man bereits im ersten Schritt eine Angabe in Minuten, die nicht weiter verfeinert werden muss.

Diese Umrechnung ist notwendig, da der Modellierer bei der Angabe eines Abstandes in Stunden oder Tagen eine gewisse Ungenauigkeit impliziert, welcher bei der Verwendung Rechnung getragen werden sollte. So wird beispielsweise bei der Angabe eines Abstandes von einem Tag eine gewisse Toleranz erwartet und nicht, dass nach exakt 24 Stunden und 0 Minuten ein Zeitfehler ausgelöst wird. Ist trotzdem einmal eine genaue Verwendung einer anderen Zeiteinheit nötig, ist dies ohne Weiteres möglich, indem zusätzlich die restlichen Zeiteinheiten mit dem Wert 0 angegeben werden. Zum Beispiel wird die Angabe „1 Tag 0 Stunden 0 Minuten“ in exakt 1440 Minuten umgerechnet.

Wird die umzurechnende Zeitdauer in einem Intervall als Unter- oder Obergrenze verwendet, so dient als ihr Wert die entsprechende Grenze aus dem Umrechnungs-Intervall. Das gesamte Vorgehen bei der Umrechnung eines Intervalls ist nochmal in Beispiel 4.5 dargestellt.

Das Intervall $[1d10h, 2d7h]$ soll in die Basis-Zeiteinheit Minuten umgerechnet werden.

- $1d10h = 34h \approx [2010min, 2069min]$
- $2d7h = 55h \approx [3270min, 3329min]$

$\Rightarrow [2010min, 3329min]$

Aus den beiden Umrechnungs-Intervallen wird jeweils der Minimal- respektive Maximal-Wert für das Ergebnisintervall $[2010min, 3329min]$ verwendet. Dieses entspricht dem Intervall $[1d9h30min, 2d7h29min]$.

Beispiel 4.5: Umrechnung eines Zeitintervalls mit unterschiedlichen Zeiteinheiten

Im Folgenden wird ohne Beschränkung der Allgemeinheit davon ausgegangen, dass sämtlich Zeit-Werte bereits in der Basis-Zeiteinheit vorliegen. Ist dies nicht der Fall, können diese wie beschrieben transformiert werden.

### 4.2.3 Transformation

Um ein *Prozessschema* in das *Zeitmodell* zu transformieren, wird zunächst der Quellknoten $E_0$ in das Zeitmodell eingefügt. Wie zuvor erwähnt, erhält dieser den Wertebereich $[0, 0]$. Danach werden die Aktivitäten transformiert. Dazu werden für jeden Knoten $K_i$ des Prozessgraphen zwei Knoten $E_{i_{start}}$ und $E_{i_{end}}$ in das Zeitmodell eingefügt, welche das

Start- und End-Ereignis des Prozessknotens repräsentieren. Diese erhalten zunächst ihr Start- beziehungsweise Endzeitintervall als Wertebereiche[3]. Ist eines dieser beiden Intervalle nicht spezifiziert, so wird dafür das Intervall $[0, \infty]$ verwendet. Als Nächstes werden die beiden Knoten $E_{i_{start}}$ und $E_{i_{end}}$ durch ein Constraint $c_{i_{start}i_{end}}$ verbunden, welches als Einschränkung die Dauer der Aktivität erhält (siehe Abbildung 4.12).

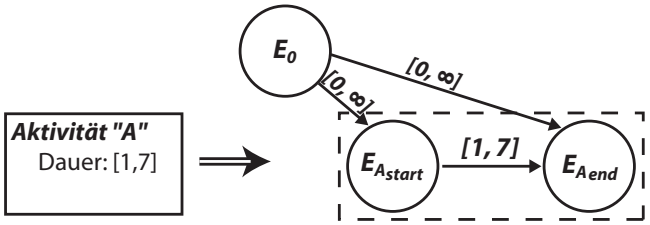



Abbildung 4.12: Transformation eines Prozessknoten

Sind alle Aktivitäten transformiert, werden für alle Kontroll-, Sync- und Zeitkanten $e$ Constraints $c$ in den Zeit-Graphen eingefügt. Dabei geben die beiden Referenzpunkte $\texttt{RP}_e^{start}$ und $\texttt{RP}_e^{dest}$ an, ob das Constraint im Zeitmodell am Start- oder End-Knoten einer Aktivität startet bzw. endet. Außerdem erhält das neue Constraint als Einschränkung die minimale und maximale Zeitdauer der entsprechenden Kante des Prozessgraphen. Das heißt, eine Kante $e = (A, B, \texttt{eT}_e, end, start, < [\texttt{Za}_e^{min}, \texttt{Za}_e^{max}], \texttt{P}_e >)$ mit $\texttt{eT}_e \in \{\text{CONTROL\_EDGE}, \text{SYNC\_EDGE}, \text{TIME\_EDGE}\}$ wird zu einem Constraint $c_{A_{end}B_{start}}$ im Zeitmodell transformiert, welches die beiden Knoten $E_{A_{end}}$ und $E_{B_{start}}$ verbindet und die Einschränkung $[\texttt{Za}_e^{min}, \texttt{Za}_e^{max}]$ repräsentiert.

Bei Schlüsselaktivitäten, das heißt bei Fristen, erhält das Constraint, welches den Quellknoten $E_0$ mit dem durch den Referenzpunkt $\texttt{RP}_n$ angegeben Event der Schlüsselaktivität verbindet, als Einschränkung den Termin der Schlüsselaktivität. Ist dieser noch unbekannt, da er erst im Verlauf der Ausführung ermittelt werden kann, behält dieses Constraint seinen bisherigen Wert als Einschränkung bei. Dieser wird dann durch den endgültigen Termin ersetzt, sobald er bekannt ist.

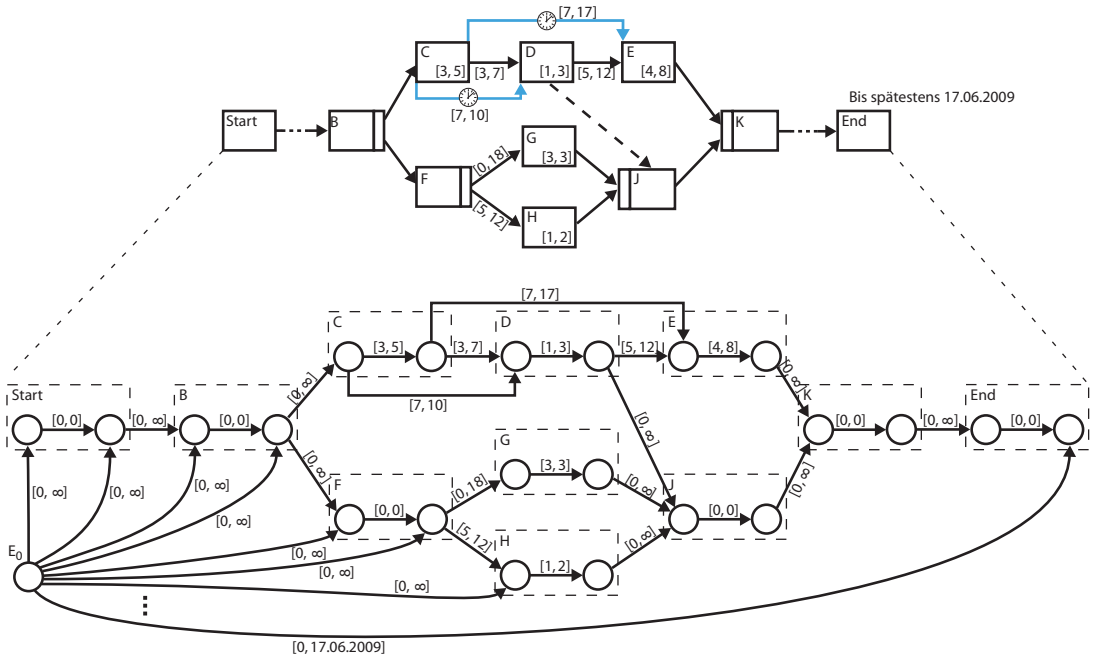
Die Transformation eines Prozessschemas unter Anwendung dieser Regeln in das Zeitmodell ist in Beispiel 4.6 zu finden.

### 4.2.4 Zeit-Algorithmen

Nachdem ein Prozessschema in ein *Zeitmodell* transformiert worden ist, muss dieses zunächst auf *Konsistenz* überprüft werden. Das heißt, es muss geprüft werden, ob es überhaupt Werte für die einzelnen Variablen gibt, welche alle Constraints zugleich erfüllen. Hierfür gibt es verschiedene Algorithmen wie beispielsweise Floyd-Warshall's *All-Pair-*

[3]Das heißt jeweils ein Constraint $c_{0i_{start}}$ bzw. $c_{0i_{end}}$ zum Quellknoten.

Die nachfolgende Abbildung zeigt die Transformation eines einfach Prozessschemas in das Zeitmodell. Dabei sind aus Gründen der Übersichtlichkeit nur einige der vom Quellknoten $E_0$ des Zeitmodells ausgehenden Constraints exemplarisch eingezeichnet.
Eine Besonderheit bei diesem Beispiel ist, dass die Spezial-Knoten des Prozessschemas die Dauer 0min besitzen, da sie automatisch vom System ausgeführt werden und daher keine nennenswerte Zeit benötigen.




Beispiel 4.6: Transformation

*Shortest-Path*-Algorithmus [Neumann & Morlock, 1993] oder Mackworth's *PC-1* und *PC-2* [Mackworth, 1975]. Ist das Zeitmodell konsistent, so kann aus diesem ein *Schedule* extrahiert werden. Dieser weist jeder Variable des Zeitmodells einen Wert zu, sodass alle Constraints des Zeitmodells erfüllt sind.

#### 4.2.4.1 Konsistenz-Algorithmus

Der hier vorgestellte Konsistenz-Algorithmus basiert auf dem in [Dechter *et al.*, 1991] für Temporal Constraint Networks adaptierten PC-2 Algorithmus aus [Mackworth, 1975].

Bevor wir den Algorithmus vorstellen, müssen noch einige Operationen und Relationen auf Constraints definiert werden, die im Rahmen des Algorithmus verwendet werden. Diese werden in Abbildung 4.13 graphisch dargestellt.

**Definition 4.3** (Negation, Schnitt, Komposition, Strenger)**.** Seien $S_{ij} = [s_1, s_2] \subseteq [-\infty, \infty]$, $T_{ij} = [t_1, t_2] \subseteq [-\infty, \infty]$ und $U_{jk} = [u_1, u_2] \subseteq [-\infty, \infty]$ zusammenhängende Constraints.

1. Die *Negation* von $S_{ij}$ ist das Constraint $S_{ji}$ zwischen $j$ und $i$ mit dem negativen Wert von $S_{ij}$
$$-S_{ij} = S_{ji} = [-s_2, -s_1]$$

2. Der *Schnitt* von $T_{ij}$ und $S_{ij}$, geschrieben $T_{ij} \cap S_{ij}$, enthält nur Werte, die sowohl in $T_{ij}$ als auch in $S_{ij}$ liegen,
$$T_{ij} \cap S_{ij} = \begin{cases} [\max(t_1, s_1), \min(t_2, s_2)] & \text{falls } \max(t_1, s_1) \leq \min(t_2, s_2) \\ \emptyset & \text{sonst} \end{cases}$$

3. Die *Komposition* von $T_{ij}$ und $U_{jk}$, geschrieben $T_{ij} \oplus U_{jk}$, enthält nur Werte $r$, für die es ein $t \in T_{ij}$ und ein $u \in U_{ij}$ gibt, sodass $t + u = r$,
$$T_{ij} \oplus U_{jk} = R_{ik} = [t_1 + s_1, t_2 + s_2]$$

4. Das Constraint $T_{ij}$ ist *strenger* als $S_{ij}$, geschrieben $T_{ij} \subseteq S_{ij}$, wenn jedes von $T_{ij}$ erlaubte Wertepaar auch von $S_{ij}$ erlaubt wird.
$$T_{ij} \subseteq S_{ij} \Leftrightarrow [t_1, t_2] \subseteq [s_1, s_2]$$

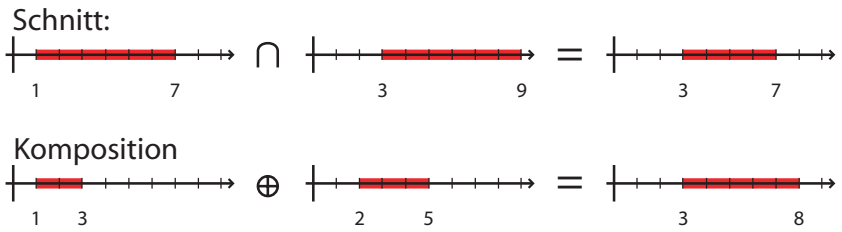



Abbildung 4.13: Operationen auf Constraints

Der PC-2 Algorithmus ist ein optimierter *Pfad-Konsistenz*-Algorithmus für Temporal Constraint Networks. Die Idee besteht darin, dass zwei Variablen $X_i$ und $X_j$ mit $X_k$ genau dann pfadkonsistent sind, wenn es für jedes Paar von Werten $(a, b) \in Dom(X_i) \times Dom(X_j)$, welches das Constraint $c_{ij}$ zwischen $X_i$ und $X_j$ erfüllt, einen Wert $c \in Dom(X_k)$ gibt, sodass $(a, c)$ und $(c, b)$ die Constraints zwischen $X_i$ und $X_k$ beziehungsweise $X_k$ und $X_j$ erfüllen (siehe Abbildung 4.14). Formal:

**Definition 4.4** (Pfad-Konsistenz [Schwalb & Dechter, 1997])**.**

- Ein Constraint $c_{ij}$ ist pfadkonsistent, genau dann wenn

$$\forall k : c_{ij} \subseteq (c_{ik} \oplus c_{kj}).$$

- Ein Constraint-Netz ist pfadkonsistent, genau dann, wenn all seine Constraints pfadkonsistent sind.

Der Algorithmus entfernt nun solange inkonsistente Werte aus den Wertebereichen der Variablen, bis das gesamte Constraint-Netz pfadkonsistent ist (siehe Abbildung 4.14). Entsteht bei der Herbeiführung der Pfad-Konsistenz eine Variable mit leerem Wertebereich, so ist das Netz inkonsistent (siehe Beispiel 4.7). Es kann also für diese Variable kein gültiger Wert gefunden werden, mit dem alle Constraints zugleich erfüllbar sind.

Im Allgemeinen ist die Pfad-Konsistenz nur eine notwendige aber keine hinreichende Bedingung für die Konsistenz eines Constraint Satisfaction Problems, da sich Inkonsistenzen auch über längere Pfade hinweg manifestieren können. Wie jedoch gezeigt werden kann [Dechter *et al.*, 1991], genügt es für STPs bereits Pfad-Konsistenz nachzuweisen, um zu zeigen, dass das gesamte Constraint-Netz konsistent ist. Der Hauptgrund hierfür liegt darin, das Simple Temporal Problems nur '<'-, '>'- und '='-Beziehungen, jedoch keine '≠'-Beziehung zwischen den Variablen erlauben.


Abbildung 4.14: Constraint Einschränkung

Der Pfad-Konsistenz-Algorithmus PC-2 nach [Schwalb & Dechter, 1997] ist in Algorithmus 1 zu finden. Hierbei handelt es sich um eine optimierte Variante des ursprünglichen Pfad-Konsistenz-Algorithmus, bei dem nur die von Änderungen betroffenen Pfade erneut überprüft werden. Dabei werden die Constraints solange weiter eingeschränkt, bis entweder eines der Constraints einen leeren Wertebereich erhält und der Algorithmus mit der Ausgabe „inkonsistent“ abbricht, oder keine weiteren Einschränkungen mehr nötig sind; d. h. die verbleibenden Constraints pfadkonsistent sind. Der PC2-Algorithmus besitzt, wie der ursprüngliche Pfad-Konsistenz-Algorithmus, eine Worst-Case Komplexität von $O(n^3)$ ($n$ Anzahl der Knoten des Zeitmodells), ist im Mittel jedoch effizienter.

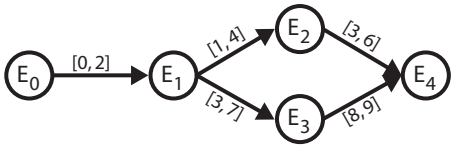
Bricht der Algorithmus seine Berechnungen mit der Ausgabe „inkonsistent“ ab, so ist es möglich, die für die Inkonsistenz verantwortlichen Constraints zu ermitteln. Hierzu werden ausgehend vom Constraint $c_{ij}$ mit $c_{ij} \cap (c_{ik} \oplus c_{kj}) = \emptyset$, bei dem der Algorithmus

```
Input : Zeitgraph (V, A, Γ) mit V = {E_0, ..., E_n} und Γ = {c_ij | 0 ≤ i, j ≤ n}
Output : Minimales Netz oder inkonsistent
1  begin
2      Q = {(i, k, j) | 0 ≤ i, k, j ≤ n ∧ (i < j) ∧ (k ≠ i, j)};
3      while Q ≠ ∅ do
4          select and delete a path (i, k, j) from Q;
5          if c_ij ≠ c_ij ∩ (c_ik ⊕ c_kj) then
6              c_ij = c_ij ∩ (c_ik ⊕ c_kj);
7              if c_ij = ∅ then return (Netzwerk inkonsistent);
8              Q = Q ∪ {(i, j, l), (l, i, j) | 0 ≤ l ≤ n, i ≠ l ≠ j};
9          end
10     end
11 end
```

**Algorithmus 1** : PC-2 Algorithmus [Schwalb & Dechter, 1997]

die Inkonsistenz erkannt hat, die Veränderungen der Constraints $c_{ij}$, $c_{ik}$ und $c_{kj}$ und der zu diesen Veränderungen führenden Constraints zurückverfolgt. Dabei genügt es für $c_{ij}$, $c_{ik}$ und $c_{kj}$ jeweils, den Veränderungen nachzugehen, die zu einer Einschränkung der zueinander näher liegenden Extremwerte der Intervalle des Schnittes führten. Das heißt, falls gilt, $\max(c_{ij}) < \min(c_{ik} \oplus c_{kj})$ so genügt es, die Einschränkungen zu betrachten, die zu einer Veränderung der Obergrenze von $c_{ij}$ oder zu einer Veränderung der Untergrenzen von $c_{ik}$ oder $c_{kj}$ geführt haben. Für $\max(c_{ik} \oplus c_{kj}) < \min(c_{ij})$ genügt entsprechend die Betrachtung der Veränderungen der Obergrenzen von $c_{ik}$ oder $c_{kj}$ und der Untergrenze von $c_{ij}$. Hat man die für die Veränderung der entsprechenden Grenzen verantwortlichen Constraints gefunden, so werden deren Veränderungen weiter zurückverfolgt, wobei die Betrachtung auf die jeweilige Grenze eingeschränkt werden kann. Auf diese Weise erhält man nach und nach eine Gruppe von Constraints, die für die gefundene Inkonsistenz verantwortlich sind (zur Verdeutlichung siehe Beispiel 4.7). Aus diesen kann die inkonsistente Stelle des Prozessschemas ermittelt werden, indem die den Constraints entsprechenden Kanten und Aktivitäten bestimmt werden. Ist die inkonsistente Stelle des Prozessschemas lokalisiert, muss diese durch Veränderung der Constraints, das heißt der Abstände und Dauern, behoben werden. Danach kann das Prozessschema erneut in einen Zeit-Graphen transformiert und dieser auf Konsistenz überprüft werden.

Hat man ein zeitkonsistentes Prozessschema gefunden, so liefert der Algorithmus ein sogenanntes *minimales Netz* als Ergebnis. Das minimale Netz $M$ eines Temporal Constraint Netzes $T$ ist ein Temporal Constraint Netz mit der gleichen Lösungsmenge wie $T$ für das gilt, dass jeder Wert der Domäne einer jeden Variable mindestens Teil einer Lösung des Netzes ist. Formal lässt sich dies wie folgt definieren:

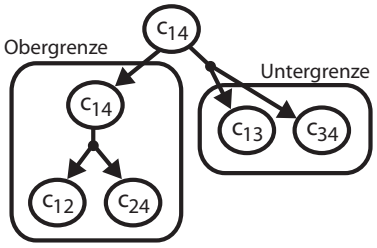



Ändert man das Zeitmodell aus Beispiel 4.4 ab und ersetzt das Constraints zwischen $E_3$ und $E_4$ durch $[8, 9]$ (siehe oben), führt Algorithmus 1 folgende Einschränkungen durch:

$c_{02} = c_{02} \cap c_{01} \oplus c_{12} = [-\infty, \infty] \cap [1, 6] = [1, 6]$
$c_{03} = c_{03} \cap c_{01} \oplus c_{13} = [-\infty, \infty] \cap [3, 9] = [3, 9]$
$c_{04} = c_{04} \cap c_{02} \oplus c_{24} = [-\infty, \infty] \cap [4, 12] = [4, 12]$
$c_{04} = c_{04} \cap c_{03} \oplus c_{34} = [4, 12] \cap [11, 18] = [11, 12]$
$c_{02} = c_{02} \cap c_{04} \oplus c_{42} = [1, 6] \cap [5, 9] = [5, 6]$
$c_{03} = c_{03} \cap c_{04} \oplus c_{43} = [3, 9] \cap [2, 4] = [3, 4]$
$c_{12} = c_{12} \cap c_{10} \oplus c_{02} = [1, 4] \cap [3, 6] = [3, 4]$
$c_{13} = c_{13} \cap c_{10} \oplus c_{03} = [3, 7] \cap [1, 4] = [3, 4]$
$c_{14} = c_{14} \cap c_{10} \oplus c_{04} = [-\infty, \infty] \cap [9, 12] = [9, 12]$
$c_{14} = c_{14} \cap c_{12} \oplus c_{24} = [9, 12] \cap [6, 10] = [9, 10]$
$c_{14} = c_{14} \cap c_{13} \oplus c_{34} = [9, 10] \cap [11, 13] = \emptyset$

$\Rightarrow$ Inkonsistenz bei $c_{14}$

Der Algorithmus stellt somit fest, dass für das Constraint $c_{14}$ eine Inkonsistenz vorliegt. Um nun herauszufinden, welche Constraints für die Inkonsistenz verantwortlich sind, werden die zuletzt durchgeführten Einschränkungen der an der Inkonsistenz beteiligten Constraints zurückverfolgt. Das heißt, für unser Beispiel genügt es, wie im Text erläutert, die Einschränkungen der Obergrenze des Constraints $c_{14}$ und der Untergrenzen der Constraints $c_{13}$ und $c_{34}$ zurück zu verfolgen. Die aktuelle Obergrenze von $c_{14}$ entstand aufgrund des Schnittes von $c_{14}$ mit der Komposition von $c_{12}$ und $c_{24}$, deren Obergrenzen bisher nicht verändert wurden. Die Untergrenzen von $c_{13}$ und $c_{34}$ wurden bisher ebenfalls nicht verändert. Man erhält damit folgenden Abhängigkeitsgraphen:




An diesem kann man nun gut erkennen, dass die Constraints $c_{13}$ und $c_{34}$ sowie die Constraints $c_{12}$ und $c_{24}$ für die Inkonsistenz verantwortlich sind.

Beispiel 4.7: Bestimmen einer Inkonsistenz

**Definition 4.5** (minimales Netz [Dechter *et al.*, 1991])**.** Seien $T$, $S$ und $M$ Zeitmodelle mit den gleichen Variablen. Dann gilt:

- Zwei Zeitmodelle $T$ und $S$ sind *äquivalent*, geschrieben $T \equiv S$, wenn sie die gleiche Lösungsmenge besitzen.
- Ein Zeitmodell $T$ ist *strenger* als ein Zeitmodell $S$, geschrieben $T \subseteq S$, wenn die '$\subseteq$'-Relation von allen einander entsprechenden Constraints erfüllt wird. Das heißt, für die Constraints $T_{ij}$ aus $T$ und $S_{ij}$ aus $S$ gilt:

$$\forall i, j : T_{ij} \subseteq S_{ij}.$$

- Das *minimale Netz* eines Zeitmodells $T$ ist das strengste Zeitmodell $M$, welches zu $T$ äquivalent ist. Das heißt $M$ ist minimales Netz von $T$ wenn gilt:

$$(M \equiv T) \wedge (\forall S : (S \equiv T) \Rightarrow (M \subseteq S))$$

Ein weiteres Nebenprodukt des PC-2 Algorithmus ist, dass dieser bei der Ermittlung des minimalen Netzes auch bisher nur implizit vorhandene Constraints ermittelt und in das minimale Netzwerk einfügt. Genauer gesagt wird für jedes Paar von Variablen ein Constraint ermittelt, welches die explizit oder implizit zwischen den beiden Variablen geltende Bedingung beschreibt. Diese können beispielsweise dazu genutzt werden, einen tieferen Einblick in die zeitlichen Zusammenhänge des Prozessschemas zu erhalten. Sie sind außerdem nötig, um zeitliche Abhängigkeiten zwischen parallel ablaufenden Aktivitäten zu erkennen, die eventuell bei der Synchronisation der beiden Aktivitäten durch eine Vereinigung entstehen. Letztere sind besonders interessant, um die entsprechenden Bedingungen durchsetzen und im Rahmen der Zeitfehlerbehandlung auch in den abhängigen Pfaden frühzeitig reagieren zu können. Beispiel 4.8 verdeutlicht diesen Sachverhalt nochmals und zeigt außerdem die Vorgehensweise des Algorithmus bei der Bestimmung des minimalen Netzes.

Hat man ein konsistentes minimales Netz für den Zeit-Graphen gefunden, so kann dieses im nächsten Schritt dazu verwendet werden, einen Schedule für die Ereignisse des Zeitmodells zu ermitteln. Dessen Werte können dann als Zeitpunkte für die Aktivitäten des Prozesses verwendet werden.

#### 4.2.4.2 Ermitteln eines Schedule

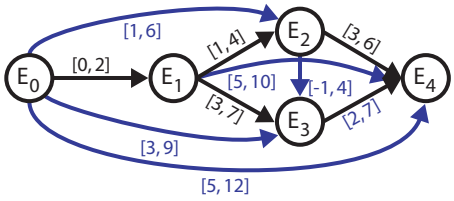
Ein *Schedule* [Bettini *et al.*, 2002b] weist jeder Aktivität $A_i$ eine Start- sowie eine Endzeitspanne $[FZ_i^{begin}, SZ_i^{begin}]$ bzw. $[FZ_i^{end}, SZ_i^{end}]$ zu, in der das jeweilige Ereignis eintreten darf, damit der Prozess unter Einhaltung aller zeitlichen Beschränkungen beendet werden kann. Außerdem gibt er noch ein, möglicherweise gegenüber dem ursprünglichen Modell eingeschränktes, Intervall $[d_i^{min}, d_i^{max}]$ für die Dauer der jeweiligen Aktivität an. Formal:

Wird Algorithmus 1 auf Beispiel 4.4 angewendet, so führt der Algorithmus in Zeile 6 folgende Einschränkungen durch:

$$
\begin{aligned}
c_{02} &= c_{02} \cap c_{01} \oplus c_{12} = [-\infty, \infty] \cap [1, 6] = [1, 6] \\
c_{03} &= c_{03} \cap c_{01} \oplus c_{13} = [-\infty, \infty] \cap [3, 9] = [3, 9] \\
c_{04} &= c_{04} \cap c_{02} \oplus c_{24} = [-\infty, \infty] \cap [4, 12] = [4, 12] \\
c_{04} &= c_{04} \cap c_{03} \oplus c_{34} = [4, 12] \cap [5, 18] = [5, 12] \\
c_{14} &= c_{14} \cap c_{10} \oplus c_{04} = [-\infty, \infty] \cap [3, 12] = [3, 12] \\
c_{14} &= c_{14} \cap c_{12} \oplus c_{24} = [3, 12] \cap [4, 10] = [4, 10] \\
c_{14} &= c_{14} \cap c_{13} \oplus c_{34} = [4, 10] \cap [5, 16] = [5, 10] \\
c_{23} &= c_{23} \cap c_{20} \oplus c_{03} = [-\infty, \infty] \cap [-3, 8] = [-3, 8] \\
c_{23} &= c_{23} \cap c_{21} \oplus c_{13} = [-3, 8] \cap [-1, 6] = [-1, 6] \\
c_{23} &= c_{23} \cap c_{24} \oplus c_{43} = [-1, 6] \cap [-6, 4] = [-1, 4] \\
c_{34} &= c_{34} \cap c_{31} \oplus c_{14} = [2, 9] \cap [-2, 7] = [2, 7]
\end{aligned}
$$

Das resultierende minimale Netz $TM = (\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$ ergibt sich wie folgt:
$\mathbf{V} = \{E_0, E_1, E_2, E_3, E_4\}$, $\mathbf{A} = \{(E_0 \rightarrow E_1), (E_1 \rightarrow E_2), (E_1 \rightarrow E_3), (E_2 \rightarrow E_4), (E_3 \rightarrow E_4)\}$ und

$$
\mathbf{\Gamma} = (c_{ij}) = \begin{bmatrix}
[0,0] & [0,2] & [1,6] & [3,9] & [5,12] \\
[-2,0] & [0,0] & [1,4] & [3,7] & [5,10] \\
[-6,-1] & [-4,-1] & [0,0] & [-1,4] & [3,6] \\
[-9,-3] & [-7,-3] & [-4,1] & [0,0] & [2,7] \\
[-12,-5] & [-10,-5] & [-6,-3] & [-7,-2] & [0,0]
\end{bmatrix}
$$




Wie ersichtlich, hat der Algorithmus zum einen das Constraint $c_{34}$ auf den Wertebereich $[2, 7]$ eingeschränkt und zum anderen sämtliche bisher nur implizit vorhandenen Constraints explizit ermittelt. Dadurch entdeckt man beispielsweise, dass zwischen den Zeitpunkten der beiden zunächst unabhängig erscheinenden Knoten $E_2$ und $E_3$ doch eine Abhängigkeit existiert. Außerdem kann man am Wertbereich des Constraints $c_{04}$ die minimale und maximale Gesamtdauer des Zeitgraphen ablesen.

Beispiel 4.8: Konsistenz-Algorithmus und minimales Netz

**Definition 4.6** (Schedule [Bettini *et al.*, 2002b])**.** Seien $A_1, \ldots, A_k$ die Aktivitäten eines Prozesses, dessen Constraints in einem Zeitmodell $TM$repräsentiert seien. Ein *Schedule* weist jeder Aktivität $A_i$ ein Tripel von Intervallen

$$([FZ_i^{begin}, SZ_i^{begin}], [d_i^{min}, d_i^{max}], [FZ_i^{end}, SZ_i^{end}])$$

zu, sodass gilt:
Wird für jede Aktivität $A_i$ ein Paar $(x_i, y_i) \in [FZ_i^{begin}, SZ_i^{begin}] \times [FZ_i^{end}, SZ_i^{end}]$, welches das Constraint $[d_i^{min}, d_i^{max}]$ erfüllt, verwendet, um die entsprechenden Paare von Anfangs- und End-Knoten von $A_i$ in $TM$zu instanzieren, so kann diese Zuweisung zu einer Lösung von $TM$erweitert werden.

Die Motivation hinter dieser Definition ist, dass die Ausführung einer Aktivität zeitlich unabhängig von anderen Aktivitäten sein sollte. Jeder Agent, der eine Aktivität ausführt, sollte nur auf die „lokalen“ Zeitbedingungen seiner Aktivität achten müssen, wobei die globalen Zeitbedingungen automatisch erfüllt sind, solange alle Agenten ihre lokalen Bedingungen erfüllen. Das heißt, wenn alle Aktivitäten in ihren vorgegebenen Intervallen starten und enden und dabei ihre vorgegebene Dauer einhalten, garantiert der Schedule, dass weiterhin alle Zeitbedingungen des Prozesses erfüllbar sind.

Die bisherige Definition eines Schedule stellt sicher, dass es zumindest einen zeitlichen Ablauf für den Prozess gibt, welcher alle Zeitbedingungen erfüllt. Sie besitzt jedoch den Nachteil, dass für manche Aktivitäten $A_i$ möglicherweise die ursprünglich spezifizierte Dauer $[D_i^{min}, D_i^{max}]$ eingeschränkt wird. Dies ist nicht immer wünschenswert bzw. ohne Weiteres möglich. Daher wird noch ein sogenannter *freier Schedule* definiert, der als zusätzliche Bedingung enthält, dass keine Einschränkung der Dauer stattfinden darf:

**Definition 4.7** (Freier Schedule [Bettini *et al.*, 2002b])**.** Ein *freier Schedule* ist ein Schedule, bei dem nur ein Zeitfenster für das Start-Ereignis einer jeden Aktivität vorgegeben ist, wobei die ursprünglich vorgegebene Dauer der Aktivität unangetastet bleibt und das End-Zeitfenster implizit ist.

Das heißt, ein Schedule

$$\langle([FZ_1^{begin}, SZ_1^{begin}], [d_1^{min}, d_1^{max}], [FZ_1^{end}, SZ_1^{end}]), \ldots,$$
$$([FZ_k^{begin}, SZ_k^{begin}], [d_k^{min}, d_k^{max}], [FZ_k^{end}, SZ_k^{end}])\rangle$$

für Aktivitäten $A_1, \ldots, A_k$, deren Constraints in einem Zeitmodell $TM$repräsentiert sind, wird als frei bezeichnet, wenn gilt:

1. Die Dauer einer jeden Aktivität $A_i$ ist dieselbe wie in $TM$, d. h. $d_i^{min} = D_i^{min}$ und $d_i^{max} = D_i^{max}$
2. Das End-Zeit-Fenster ist implizit: $FZ_i^{end} = FZ_i^{begin} + d_i^{min}$ und $SZ_i^{end} = SZ_i^{begin} + d_i^{max}$

Leider besitzt nicht jeder Prozess mit konsistentem Zeitmodell einen freien Schedule. Wenn beispielsweise bei der Gewinnung des minimalen Netzes ein Constraint, welches eine Dauer repräsentiert, verändert wurde, so kann kein freier Schedule mehr existieren, da eine der angegebenen minimalen oder maximalen Dauern nicht mehr Teil irgendeines Schedules sein kann. Doch selbst wenn dies nicht der Fall ist, kann nicht garantiert werden, dass ein freier Schedule existiert (siehe Beispiel 4.9). Da ein freier Schedule jedoch Voraussetzung dafür ist, dass bei der Ausführung der Aktivitäten die angegebenen Dauern in vollem Umfang ausgenützt werden können, wird die Konsistenz der Zeitbedingungen eines Prozessschemas wie folgt definiert:

**Definition 4.8** (Konsistenz)**.** Die Zeitbedingungen eines Prozessschemas sind genau dann konsistent, wenn das Prozessschema mindestens einen freien Schedule besitzt.

Unter Umständen ist es möglich, diese relativ strenge Definition der Konsistenz zu lockern. Eine solche Relaxierung ist z. B. realisierbar, wenn es nicht unbedingt notwendig ist, dass die vollständige Dauer einer Aktivität ausgenützt werden kann, etwa wenn diese „großzügig“ gewählt wurde. Wie mit solchen Fällen verfahren werden kann, zeigen wir in Abschnitt 4.2.6.

Um nun aus dem minimalen Netz $M$ eines Zeitmodells $TM$einen freien Schedule zu gewinnen wird aus den Daten des minimalen Netzes ein neuer, reduzierter Zeitgraph erstellt. Dieser repräsentiert die Menge aller freien Schedules mit einem einzelnen Punkt als Startzeitfenster jeder Aktivität. Ein solcher Schedule umfasst die Information

$$([x_i, x_i], [D_i^{min}, D_i^{max}], [x_i + D_i^{min}, x_i + D_i^{max}])$$

für jede Aktivität, wobei $[D_i^{min}, D_i^{max}]$ die Werte für die Dauer der Aktivität $A_i$ aus dem ursprünglichen Zeitmodell sind. Die Werte für die $x_i$ müssen die Folgenden aus dem minimalen Netz $M = (\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$ mit $\mathbf{V} = \{E_0, E_{1_{start}}, E_{1_{end}}, \ldots, E_{k_{start}}, E_{k_{end}}\}$ abgeleiteten Bedingungen erfüllen [Bettini *et al.*, 2000]:

1. Für jede Aktivität $A_i$ gilt, dass die Dauer im minimalen Netz $M$ gegenüber dem Zeitmodell $TM$nicht verändert wurden, d. h.

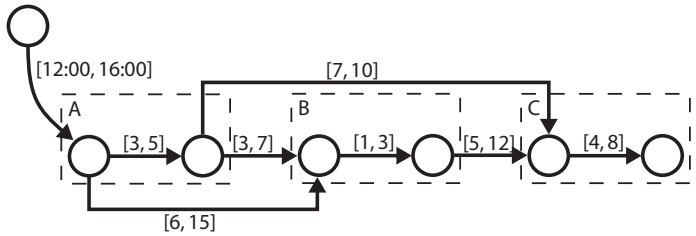
   $$c_{i_{start}i_{end}} = [D_i^{min}, D_i^{max}] \text{ mit } c_{i_{start}i_{end}} \in \Gamma.$$

2. Für jede Aktivität $A_i$ gilt:

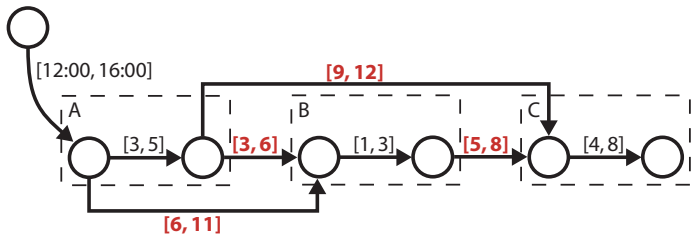
   $$x_i \in c_{0i_{start}} \text{ mit } c_{0i_{start}} \in \Gamma$$

   d. h. der gewählte Startzeitpunkt liegt im durch das minimale Netz $M$ vorgegebenen Intervall.

Bei dem dargestellten Zeitmodell eines Prozessschemas muss aufgrund des Maximal-Abstandes zwischen dem Ende der Aktivität $A$ und dem Start der Aktivität $C$ sowie der Minimal-Abstände zwischen $A \rightarrow B$ und $B \rightarrow C$ die mögliche Dauer von Aktivität $B$ reduziert werden. Daher existiert für dieses Zeitmodell zwar ein minimales Netz, aber kein freier Schedule, da für Aktivität $B$ nicht deren maximale Dauer zur Verfügung steht.




Wird der Maximal-Abstand zwischen $A$ und $C$ auf 12 Minuten erweitert, muss bei der Berechnung des minimalen Netzes (siehe unten) keine der Dauern der Aktivitäten eingeschränkt werden. Es existiert jedoch weiterhin kein freier Schedule, da, falls die Aktivität $A$ 3 Minuten benötigt, der Abstand zwischen $A$ und $B$ mindestens 2 Minuten betragen muss, um den minimal Abstand zwischen $A$ und $B$ zu erfüllen. Dies ist jedoch nicht mit der maximalen Dauer von $B$ und dem Maximal-Abstand zwischen $A$ und $C$ in Einklang zu bringen.




Wird der Maximal-Abstand zwischen dem Ende der Aktivität $A$ und dem Start der Aktivität $C$ weiter auf 13 Minuten erweitert, existiert für das Zeitmodell folgende Menge von freien Schedules:

| Aktivität | Startzeitfenster | Dauer | Endzeitfenster |
|---|---|---|---|
| Aktivität A | [12:00, 16:00] | [3min, 5min] | [12:03, 16:05] |
| Aktivität B | [12:08, 16:08] | [1min, 3min] | [12:09, 16:11] |
| Aktivität C | [12:16, 16:21] | [4min, 8min] | [12:20, 16:24] |

Beispiel 4.9: Freie Schedules

3. Für jedes Paar von Aktivitäten $\langle A_i, A_j \rangle$ mit $i < j$ muss gelten:

$$x_i + k_{ij} \leq x_j \leq x_i + K_{ij}.$$

Dabei lassen sich die Werte $k_{ij}$ und $K_{ij}$ folgendermaßen aus dem minimalen Netz $M$ berechnen:

$$k_{ij} = \max \left\{ \begin{array}{l} D_i^{max} - D_j^{min} + \min(c_{i_{end}j_{end}}), \\ \min(c_{i_{start}j_{start}}), \\ \min(c_{i_{start}j_{end}}) - D_j^{min}, \\ D_i^{max} + \min(c_{i_{end}j_{start}}) \end{array} \right\} \tag{4.1}$$

und

$$K_{ij} = \min \left\{ \begin{array}{l} D_i^{min} - D_j^{max} + \max(c_{i_{end}j_{end}}), \\ \max(c_{i_{start}j_{start}}), \\ \max(c_{i_{start}j_{end}}) - D_j^{max}, \\ D_i^{min} + \max(c_{i_{end}j_{start}}) \end{array} \right\} \tag{4.2}$$

Dabei sind $\min(c_{ij})$ und $\max(c_{ij})$ jeweils der kleinste beziehungsweise größte Wert des Constraints $c_{ij}$ und $D_i^{min}$, $D_i^{max}$, $D_j^{min}$ und $D_j^{max}$ jeweils die minimale beziehungsweise maximale Dauer der Aktivitäten $A_i$ und $A_j$.

Aus diesen Bedingungen ergibt sich ein neues Zeitmodell, dessen minimales Netz die Lösungsmenge aller freien Schedules repräsentiert. Um dieses zu ermitteln, kann beispielsweise Algorithmus 1 aus dem vorhergehenden Abschnitt verwendet werden. Gibt es für das Zeitmodell kein minimales Netz, so existiert auch kein freier Schedule für das ursprüngliche Zeitmodell, und die gesamte Zeitberechnung wird mit der Ausgabe „inkonsistent“ abgebrochen. Dabei lässt sich wieder durch eine Analyse der zuletzt veränderten Constraints ermitteln, welcher Bereich des Prozessschemas für die Inkonsistenz verantwortlich ist (siehe entsprechende Diskussion in Abschnitt 4.2.4.1 auf Seite 84).

Wählt man aus der Domäne einer Variablen der Lösungsmenge der freien Schedules einen Wert aus, so garantieren die Bedingungen, dass diese Belegung zu einem freien Schedule erweitert werden kann. Indem man die Domäne der Variable durch den ausgewählten Wert ersetzt und für das geänderte Zeitmodell erneut das minimale Netz bestimmt, kann man die Lösungsmenge der freien Schedules unter der getroffenen Einschränkung ermitteln. Dies ist besonders zur Laufzeit des Prozesses interessant. Hier kann nach dem Start oder Ende jeder Aktivität die Lösungsmenge für die freien Schedules mit den realen Werten aktualisiert werden. Damit ist es möglich neue Ausführungsintervalle für die verbleibenden Aktivitäten derart zu bestimmen, dass die Existenz eines freien Schedule weiterhin garantiert ist. Die genaue Vorgehensweise wird in Kapitel 6 erläutert.

Ein *freier Schedule* bedeutet nicht, dass eine Aktivität nur im angegebenen Intervall gestartet werden kann; er bedeutet jedoch umgekehrt, dass, solange alle vorhergehenden Bedingungen eingehalten werden, die Aktivität auf alle Fälle zu jedem Zeitpunkt des angegebenen Intervalls gestartet werden kann. Meist wird die untere Schranke des Startintervalls einer Aktivität zur Laufzeit kleiner werden, da vorhergehende Aktivitäten nicht ihre maximale Dauer benötigt haben. Diese Eigenschaft bringt auch Einschränkungen für die Zeitmodellierung mit sich, die in Kapitel 5 anhand verschiedener Beispiele vorgestellt und erläutert werden. Der große Vorteil der freien Schedules ist jedoch, dass sie garantieren, dass alle Aktivitäten ihre spezifizierte Minimal- und Maximal-Dauer benötigen dürfen und, solange alle Bedingungen erfüllt werden, im für sie berechneten Intervall ausführbar sind. Letzteres ist vor allem für das Ressourcenmanagement und im Rahmen der Vorhersage von Zeitdaten und damit für die persönlichen Terminpläne der beteiligten Agenten besonders interessant.

### 4.2.5 Alternativ-Verzweigungen

Wie in Abschnitt 4.1.6 diskutiert, gibt es verschiedene Wege, um mit den Problemen umzugehen, die sich im Zusammenhang mit Alternativ-Verzweigungen und Schleifen ergeben. Für zwei der dort vorgeschlagenen Alternativen wird nun untersucht, wie diese mit dem Zeitmodell und dessen Algorithmen kombiniert werden können.

#### 4.2.5.1 Pessimistischer Ansatz

Beim pessimistischen Ansatz werden Alternativ-Verzweigungen auf die gleiche Weise wie Parallel-Verzweigungen behandelt. Daher ist der pessimistische Ansatz bereits größtenteils durch den in Abschnitt 4.2.4 vorgestellten Algorithmus 1 und den Bedingungen für die Lösungsmenge der freien Schedules erfüllt. Es gilt an dieser Stelle lediglich festzulegen, wie mit Schleifen verfahren werden soll. Betrachtet man die Eigenschaften des pessimistischen Ansatzes genauer, stellt man fest, dass eine Berechnung über mehrere Iterationen einer Schleife in sehr vielen Fällen dazu führt, dass das Prozessschema als inkonsistent erkannt wird. Daher wird für den pessimistischen Ansatz immer nur jeweils der erste Durchlauf bzw. die minimale Anzahl an Durchläufen einer Schleife in das Zeitmodell eingefügt. Während der Ausführung des Prozesses wird dann, sobald eine Schleifeniteration beendet ist und eine neue beginnt, eine weitere Iteration der Schleife in das Zeitmodell eingefügt. Um dies zu vereinfachen und auch um der Unbestimmtheit der Anzahl von Iterationen Rechnung zu tragen, bekommt das eine Schleife verlassende, genauer gesagt das vom ENDLOOP-Knoten ausgehende Constraint im Zeitmodell immer den Wertebereich $[Za_{ij}^{min}, \infty]$. Abbildung 4.15 verdeutlicht diese Vorgehensweise nochmals. Es ist damit allerdings nicht mehr möglich, eine Aussage über die der Schleife folgenden Aktivitäten zu treffen.

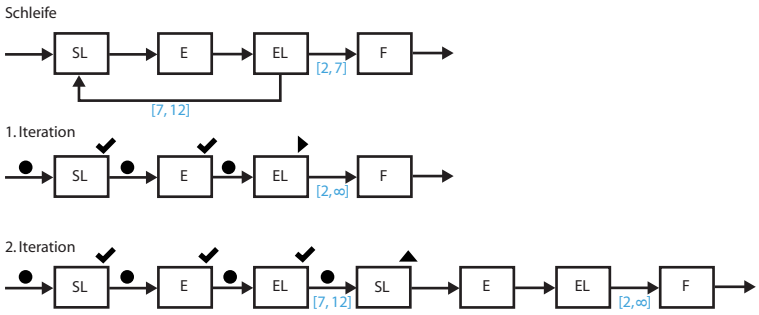



Abbildung 4.15: Schleifen im pessimistischen Ansatz

Für die Zeit-Algorithmen stellt diese Art der Behandlung von Schleifen aufgrund ihres Aufbaus kein Problem dar. Wird eine weitere Iteration der Schleife in einen Zeitgraphen eingefügt, so kann ein Großteil der auf dem „alten“ Zeitgraphen durchgeführten Berechnungen wiederverwendet werden, indem die Veränderungen am bereits berechneten minimalen Netz vorgenommen werden. Aus diesem muss dann lediglich erneut mittels Algorithmus 1 ein minimales Netz berechnet werden, wobei die initiale Menge an zu betrachtenden Pfaden auf die durch die Veränderung beeinflussten Pfade beschränkt werden kann (siehe Abschnitt 6.2.1).

#### 4.2.5.2 Unfolded Workflow Graph

Im Fall des Unfolded-Workflow-Graph-Ansatzes finden vor der Transformation in das Zeitmodell zunächst zwei Zwischentransformationen statt, die das Prozessschema $G$ in ein aufgefaltetes Prozessschema $U$ transformieren.

Dazu werden zunächst sämtliche Schleifen in eine Kaskade von XOR-Konstrukten mit jeweils einem Ausführungs-Pfad für die unterschiedlichen Instanztypen aufgefaltet. Dabei wird angenommen, dass sowohl die minimale als auch die maximale Interations-Anzahl der Schleife bekannt ist. Diese können beispielsweise aus der Historie gewonnen oder von Domänenexperten erfragt und in den Meta-Daten des ENDLOOP-Knotens hinterlegt werden. Um nun eine Schleife in eine Kaskade von XOR-Konstrukten zu transformieren, wird zunächst der Schleifenrumpf so oft kopiert, wie die minimale Iterations-Anzahl der Schleife. Die einzelnen Kopien werden dann durch Kontrollkanten verbunden, wie in Abbildung 4.16 illustriert. Als Abstandswert erhält diese zusätzliche Kontrollkante den Abstand zwischen zwei Durchläufen der Schleife, welcher durch eine Zeitkante vom Start-Ereignis des ENDLOOP- zum End-Ereignis den STARTLOOP-Knoten modelliert wurde. Zeitkanten, die sich auf Knoten derselben Iteration beziehen, werden beim Kopiervorgang einfach mit kopiert. Zeitkanten, die auf verschiedene Iterationen verweisen, werden

ebenfalls kopiert, jedoch werden ihr Anfang und Ende entsprechend der Auffaltung der Schleife angepasst (siehe Abbildung 4.16). Unterscheidet sich die maximale Anzahl an Iterationen von der minimalen, so wird die letzte Kopie des ENDLOOP-Knotens zu einem XOR-Split-Knoten (die Bedingung für eine weitere Iteration wird dabei entsprechend angepasst) und zusätzlich wird ein XOR-Join-Knoten eingefügt, welcher direkt mit dem Split Knoten verbunden wird. Dann wird ein weiterer Pfad zwischen XOR-Split und -Join eingefügt, für den der Schleifenrumpf ein weiteres Mal kopiert und entsprechend verbunden wird. Dieses Vorgehen wird so oft wiederholt, bis ein Pfad für die maximale Anzahl an Iterationen vorhanden ist. Das Ergebnis einer solchen Transformation ist in Abbildung 4.16 zu sehen.

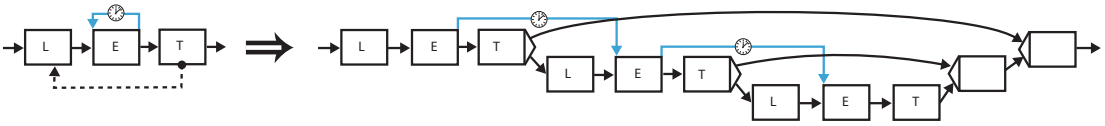



Abbildung 4.16: Transformation einer Schleife mit 1-3 Iterationen

Das so entstandene Prozessschema enthält nun keine Schleifen mehr. Somit kann im nächsten Schritt eine weitere Zwischen-Transformation stattfinden, die zusätzlich die XOR-Konstrukte des Prozessschemas auffaltet. Diese funktioniert nach [Eder *et al.*, 2000] wie in Algorithmus 2 beschrieben. Ein Beispiel hierfür sieht man in Abbildung 4.10 auf Seite 75.

Nach diesen beiden Auffaltungsschritten findet eine normale Transformation des aufgefalteten Prozessschemas $U$ in das Zeitmodell statt. Dies gelingt jetzt ohne Einschränkungen, da der neue Graph $U$ weder Schleifen noch XOR-Join-Knoten enthält und damit keine verschiedenen Instanztypen mehr zusammengefasst werden. Auf Basis des entstandenen Zeitgraphen kann nun die Zeitrechnung durchgeführt werden. Dazu muss jedoch der zuvor verwendete Algorithmus 1 an den neuen Graphen angepasst werden. Es muss bei der Auswahl der zu berechnenden Pfade darauf geachtet werden, dass alle Knoten eines Pfades zum selben Instanztyp gehören. Ähnliches gilt in Hinblick auf die Bedingungen für die Lösungsmenge der freien Schedules. Hier muss Bedingung 2. dahin gehend angepasst werden, dass die beiden Aktivitäten $A_i$ und $A_j$ zur selben Instanz gehören. Eine angepasste Variante von Algorithmus 1 bietet Algorithmus 3. Dieser verwendet die Funktion $InSameInstance$, die feststellt, ob alle drei übergebenen Knoten zum selben Instanztyp gehören. Ermittelt werden kann dies beispielsweise in dem überprüft wird, ob alle drei Knoten mindestens einen gemeinsamen Endknoten als Nachfolger im aufgefalteten Prozessgraphen besitzen (siehe Abbildung 4.17).

**Input** : Prozessschema $G = (\mathbf{N}_G, \mathbf{E}_G, \mathbf{D}_G, \mathbf{DF}_G)$
**Output** : Unfolded Workflow-Graph $U = (\mathbf{N}_U, \mathbf{E}_U, \mathbf{D}_U, \mathbf{DF}_U)$
**begin**

Um einen Prozessschema $G$ in einen Unfolded Workflow-Graphen $U$ zu transformieren sind folgende Schritte nötig:

1. Der Startknoten von $G$ wird nach $U$ kopiert.
2. Die Knoten von $G$ werden in topologischer Reihenfolge besucht. Dabei wird für jeden Knoten $n \in \mathbf{N}_G$ überprüft, ob er mehr als einen Vorgänger in $U$ besitzt:
   - Falls $|\{n' \in \mathbf{N}_U | n' \prec_{Control\_e \cup Sync\_e \cup Time\_e} n\}| = 1$ so werden für diesen Knoten so viele Kopien in $U$ eingefügt, wie es Kopien des Vorgängers von $n$ gibt. Danach werden diese Knoten so verbunden, dass jede Kopie von $n$ mit genau einer Kopie des Vorgängers von $n$ verbunden ist und umgekehrt.
   - Falls $|\{n' \in \mathbf{N}_U | n' \prec_{Control\_e \cup Sync\_e \cup Time\_e} n\}| \geq 1$ so werden so viele Kopien von $n$ in $U$ eingefügt, wie es gültige Vorgänger-Kombinationen von $n$ gibt. Diese Kombinationen werden berechnet, in dem alle Kombinationen von Kopien von Vorgängern von $n$ in $G$ erzeugt werden, sodass es in jeder Kombination genau eine Kopie eines jeden Vorgängers gibt. Das heißt, es wird das kartesische Produkt der Kopien der entsprechenden Knoten gebildet. Dann werden diese Knoten entsprechend mit den Knoten ihrer jeweiligen Kombination verbunden. Kopien von XOR-Join-Knoten in $G$ haben jetzt noch genau einen Vorgänger in $U$ und sind daher keine Join-Knoten mehr.

**end**

**Algorithmus 2** : Unfolding-Prozedur [Eder *et al.*, 2000]

**Input** : Zeitgraph $(\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$ mit $\mathbf{V} = \{E_0, \ldots, E_n\}$ und $\mathbf{\Gamma} = \{c_{ij} | 0 \leq i, j \leq n\}$
**Output** : Minimales Netz oder inkonsistent

```
1  begin
       Q = {(i,k,j)|0 ≤ i,k,j ≤ n ∧ (i < j) ∧ (k ≠ i,j)
2                  ∧ InSameInstance(E_i, E_k, E_j)};
3      while Q ≠ ∅ do
4          select and delete a path (i,k,j) from Q;
5          if c_ij ≠ c_ij ∩ c_ik ⊕ c_kj then
6              c_ij = c_ij ∩ (c_ik ⊕ c_kj);
7              if c_ij = ∅ then return (Netzwerk inkonsistent);
               Q = Q ∪ {(i,j,l),(l,i,j)|0 ≤ l ≤ n ∧ (i ≠ l ≠ j)
8                                  ∧ InSameInstance(E_i, E_k, E_j)};
9          end
10     end
11 end
```

**Algorithmus 3** : Erweiterter PC-2 Algorithmus [Schwalb & Dechter, 1997] (modifiziert)

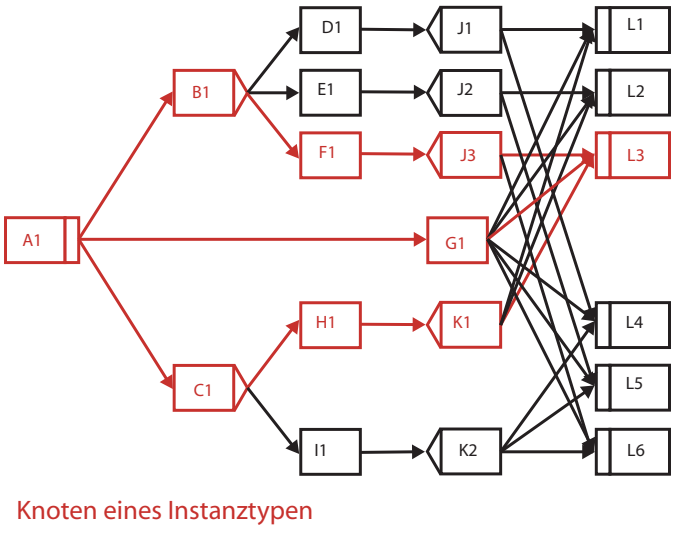



Abbildung 4.17: Knoten eines Instanztypen

### 4.2.6 Relaxierung von Zeitbedingungen

Es ist nicht immer notwendig, dass einem Bearbeiter die spezifizierte Maximal-Dauer für eine Aktivität auch zur Verfügung steht. Beispielsweise wird diese oftmals zu groß angegeben, um auch eventuell unvorhergesehene Fälle mit abdecken zu können. Führt dies dann zu einem inkonsistenten Prozessschema, nützt selbst die sicherste Zeitplanung nichts, wenn die angegeben Werte niemals eingehalten werden können. An dieser Stelle kann das System bei der Bestimmung eines Schedules von den Anforderungen des *freien Schedules* abweichen, um ein konsistentes Prozessschema zu erzielen.

Das System ist jedoch nicht selbst in der Lage, zu entscheiden, welche Aktivitäten es einschränken kann und wie weit dies geschehen darf. Um also eine Automatisierung dieser Einschränkung bieten zu können, muss der Zeitparameter der Aktivitäten um einen weiteren Wert $D^{minmax}$ ergänzt werden, welcher angibt, inwieweit die maximale Dauer vom System eingeschränkt werden darf. Mithilfe dieses Wertes lässt sich dann in Ergänzung zum in Abschnitt 4.2.4.2 definierten freien Schedule ein sogenannter *beschränkter Schedule* definieren:

**Definition 4.9** (Beschränkter Schedule (angelehnt an [Bettini *et al.*, 2002b]))**.** Ein *beschränkter Schedule* ist ein Schedule, bei dem ein Zeitfenster für das Start-Ereignis einer jeden Aktivität vorgegeben ist, die maximale Dauer möglicherweise gegenüber der ursprünglich vorgegebenen maximalen Dauer der Aktivität eingeschränkt ist, die minimale Dauer unangetastet bleibt und das End-Zeitfenster implizit ist.

Ein Schedule

$$\langle([FZ_1^{begin}, SZ_1^{begin}], [d_1^{min}, d_1^{max}], [FZ_1^{end}, SZ_1^{end}]), \ldots$$
$$([FZ_k^{begin}, SZ_k^{begin}], [d_k^{min}, d_k^{max}], [FZ_k^{end}, SZ_k^{end}])\rangle$$

für Aktivitäten $A_1, \ldots, A_k$, deren Constraints in einem Zeitmodell $TM$repräsentiert sind, wird als beschränkt bezeichnet, wenn gilt:

1. Die maximale Dauer einer jeden Aktivität $A_i$ ist möglicherweise gegenüber der in $TM$angegebenen maximalen Dauer bis auf nicht weniger als $D_i^{minmax}$ eingeschränkt, die minimale Dauer ist dieselbe wie in $TM$, d. h. $d_i^{min} = D_i^{min}$ und $D^{minmax} \leq d_i^{max} \leq D_i^{max}$
2. $FZ_i^{end} = FZ_i^{begin} + d_i^{min}$ und $SZ_i^{end} = SZ_i^{begin} + d_i^{max}$

Um aus einem minimalen Netzwerk einen solchen beschränkten Schedule zu gewinnen, können die Bedingungen für einen freien Schedule aus Abschnitt 4.2.4.2 übernommen werden. Lediglich die erste Bedingung, dass die ursprüngliche maximale Dauer im minimalen Netzwerk nicht eingeschränkt sein darf, wird zugunsten der Bedingung fallen gelassen werden, dass die neue maximale Dauer nicht kleiner als $D_i^{minmax}$ sein darf. Das heißt Bedingung 1 lautet nun wie folgt:

1.' Für jede Aktivität $A_i$ gilt, dass die minimale Dauer im minimalen Netz gegenüber dem Zeitmodell nicht verändert, und die maximale Dauer nicht auf weniger als $D_i^{minmax}$ eingeschränkt wurde, d. h.

$$[D_i^{min}, D_i^{minmax}] \subseteq c_{i_{start}i_{end}} \subseteq [D_i^{min}, D_i^{max}]$$

mit $c_{i_{start}i_{end}} \in \mathbf{\Gamma}$ und $M = (\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$ ist das minimale Netz.

Die vorgeschlagene Variante betrachtet nicht alle beschränkten Schedules, sondern nur die, gegenüber den durch das minimale Netz gemachten Einschränkungen, freien Schedules. Das Betrachten aller beschränkten Schedules führt zu einem linearen Optimierungsproblem, da die Bedingungen für die Lösungsmenge dann Variablen für die maximale Dauer enthalten. Eine ausführlichere Diskussion dieser Thematik ist in [Bettini *et al.*, 2002b] zu finden.

Es ist auch möglich, den Algorithmen zu erlauben, die minimale Dauer einer Aktivität einzuschränken. Dies ist jedoch nur wenig sinnvoll, da es dem Benutzer künstliche Zeiteinschränkungen auferlegt und ihn dazu zwingt, mit dem Beenden einer Aktivität zu warten, bis die entsprechende Zeitspanne verstrichen ist. Eine ausführlichere Diskussion dieser sogenannten *eingeschränkten Schedules* findet sich ebenfalls in [Bettini *et al.*, 2002b].

## 4.3 Zusammenfassung

Das ADEPT2-Basismodell ist, wie sich nochmals gezeigt hat, gut für eine Erweiterung um die geforderten Zeitkonstrukte geeignet. Vor allem die zugrunde liegende Blockstruktur ermöglichte eine klare syntaktische und semantische Definition der neu hinzugefügten Zeitkanten. Lediglich bei Schleifen müssen hier einige Besonderheiten beachtet werden. Etwas aufwendiger ist die Unterstützung dynamischer Zeitbedingungen, doch auch hier lies sich eine konsistente Lösung finden.

Zusätzliche Zeitzustände erweitern den Ausführungszustand einer Aktivität und helfen dem Prozess-Management-System beim Umgang mit den Zeitdaten der Prozessschemata. Ein noch offenes Problem ist die Behandlung verschiedener Instanztypen. Hierfür wurden verschiedene Möglichkeiten vorgestellt, die je nach Anforderung an das PMS und die Prozesse zum Einsatz kommen können.

Um auf einfachere Weise mit den Zeitbedingungen eines Prozessschemas umgehen und Berechnungen auf diesen durchführen zu können, wurde ein Zeitmodell entwickelt. Dieses basiert auf Temporal Constraint Networks und ermöglicht damit eine konsistente Zeitrechnung mittels wohldefinierter Formalismen. Darauf aufbauend wurde zunächst untersucht, wie die zeitliche Konsistenz eines Prozessschemas sichergestellt werden kann und wie man, falls nötig, inkonsistente Stelle des Prozessschemas lokalisieren kann. Außerdem wurde demonstriert, wie aus dem Zeitmodell Zeitpunkte für die Aktivitäten

ermittelt werden können, zu denen eine Ausführung der Aktivitäten, unter Einhaltung aller Zeitbedingungen, möglich ist.

Bei den bisherigen Betrachtungen des Zeitmodells waren verschiedene Instanztypen ausgeschlossen. Deshalb wurde für zwei der zuvor vorgeschlagenen Möglichkeiten zur Behandlung verschiedener Instanztypen eine Kombination mit dem entwickelten Zeitmodell vorgestellt.

Abschließend wurde diskutiert, wie im Fall eines inkonsistenten Prozessschemas, durch eine Beschränkung der Aktivitätsdauern, ein konsistentes Prozessschema mit entsprechenden Zeitpunkten für die Aktivitäten erreicht werden kann.

Im nächsten Kapitel werden nun einige Besonderheiten diskutiert, die sich aus dem gewählten Weg für die Integration der Zeitbedingungen und den gewählten Algorithmen für Modellierung von Prozessen ergeben. Das Hauptaugenmerk wird dabei auf der Modellierung der zeitlichen Ebene eines Prozesses liegen.

# 5 Ausgewählte Aspekte der Zeitmodellierung eines Prozesses

Nachdem die Grundlagen für die Zeitmodellierung geschaffen und die entsprechenden Algorithmen entwickelt worden sind, sollen nun die Abläufe bei der Modellierung eines Prozesses genau untersucht werden. Dazu sollen anhand eines Beispiels die unterschiedlichen Aspekte genauer betrachtet werden, die bei der Modellierung der Zeitbedingungen eines Prozesses eine Rolle spielen.

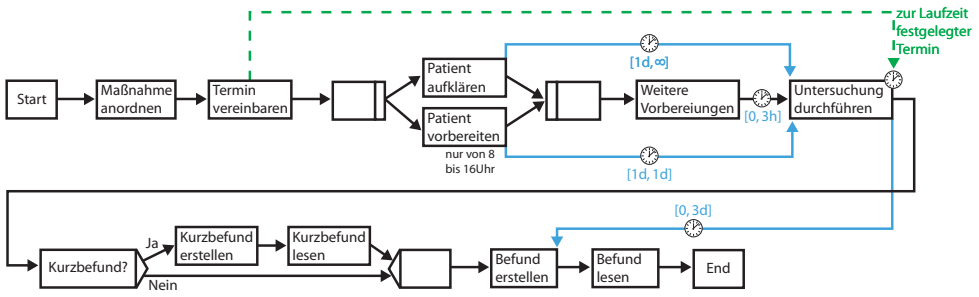
## 5.1 Vorstellung und Diskussion des Beispiel-Prozesses

Zur Demonstration der unterschiedlichen zeitlichen Aspekte soll ein Prozess aus dem klinischen Bereich dienen [Dadam *et al.*, 1995], [Konyen *et al.*, 1996], [Grimm, 1997], welcher in Beispiel 5.1 beschrieben und illustriert ist. Dieses Beispiel wurde mit aus dem Grund gewählt, dass klinische Abläufe teilweise eine große Herausforderung nicht nur für das Zeitmanagement von Prozess-Management-Systemen darstellen [Dadam *et al.*, 2000].

Ein Faktor, der eine zeitliche Abstimmung besonders wichtig macht, ist die Zusammenarbeit von verschiedenen, organisatorisch unabhängigen Bereichen sowie verschiedenen Berufsgruppen innerhalb einer Klinik. Daher müssen die einzelnen Termine aufeinander abgestimmt werden, um einen zügigen Ablauf der Prozesse gewährleisten zu können. Zusätzlich wird die Situation in der Praxis noch dadurch verkompliziert, dass häufig von einer Stelle mehrere Untersuchungen für einen Patienten angefordert werden. Diese können jedoch meist nicht in beliebiger Reihenfolge durchgeführt werden, da beispielsweise verabreichte Kontrastmittel sehr lange im Körper verbleiben und die Ergebnisse nachfolgender Untersuchungen verfälschen können. Als Konsequenz müssen bei einer Verschiebung eines Termins häufig auch andere Termine des Patienten verschoben werden. Sind die unterschiedlichen Untersuchungen innerhalb eines gemeinsamen Prozessschemas modelliert, wird bei der Verschiebung eines Termins aufgrund der Zeitbedingungen ein Zeitfehler ausgelöst, bei dessen Behandlung die Termine der anderen Untersuchungen verschoben werden müssen. Ist jedoch für jede der Untersuchungen ein eigenes *Prozessschema* spezifiziert, ist das Zeitmanagement nicht in der Lage die Abhängigkeiten zwischen den unterschiedlichen Prozessschemata zu erkennen. Sollte in diesem Fall ein Termin verschoben werden, müssen die Prozessinstanzen von der anfordernden Stelle entsprechend aktualisiert werden. Es besteht theoretisch auch die Möglichkeit, Abhängig-

keiten zwischen einzelnen Prozessschemata zu spezifizieren [Heinlein, 2001], jedoch ist dieses Thema im Rahmen dieser Arbeit nicht von Bedeutung.

Der hier verwendete beispielhafte Ablauf stellt die Vorgänge bei der Durchführung einer Untersuchung dar. Zunächst ordnet ein Arzt auf Station oder in der Ambulanz eine Untersuchung an[a]. Hierzu muss ein Termin zwischen anfordernder und untersuchender Stelle vereinbart werden.
Einen Tag vor Durchführung der Untersuchung muss der Patient vom Stationsarzt über die durchzuführende Untersuchung und deren Risiken aufgeklärt und seine schriftliche Einwilligung eingeholt werden. Außerdem muss der Patient eventuell 24 Stunden vorher auf die entsprechende Untersuchung vorbereitet werden (z. B. Verabreichen eines Kontrastmittels).
Am Tag der Untersuchung selbst sind noch weitere vorbereitende Maßnahmen nötig, etwa ein Wechsel der Kleidung oder das Verabreichen eines Beruhigungsmittels. Drei Stunden später kann die Untersuchung durchgeführt werden.
Wurde ein Kurzbefund angefordert (zum Beispiel in einem dringlichen Fall), bekommt der Patient diesen direkt nach der Untersuchung für den Stationsarzt mitgegeben. In jedem Fall aber wird der anfordernden Stelle nach maximal 3 Tagen ein ausführlicher Befund zugestellt.




[a] Aktivitäten werden durch eine Serifen-lose Schrift hervorgehoben

Beispiel 5.1: Medizinische Untersuchung [Dadam *et al.*, 1995]

## 5.2 Dauern und Zeitbedingungen

Lässt man für das im Beispiel spezifizierte Prozessschema nun die Zeitbedingungen berechnen, so bricht der Algorithmus mit der Ausgabe „inkonsistent“ ab. Macht man sich, wie auf Seite 84 beschrieben, auf die Suche nach der Ursache für die Inkonsistenz, so erhält man als Ergebnis, dass der Konsistenz-Algorithmus versucht hat, die maximale Dauer der Aktivität Weitere Vorbereitungen einzuschränken. Der Grund hierfür wiederum

ist unter anderem, dass zwischen den Aktivitäten Patient vorbereiten und Untersuchung durchführen zwar ein maximaler Abstand von einem Tag existiert, die Aktivität Weitere Vorbereitungen aber nach der aktuellen Spezifikation beliebig viel Zeit benötigen könnte. Somit kann der Algorithmus keinen Schedule für das Prozessschema bestimmen ohne die maximale Dauer von Weitere Vorbereitungen einzuschränken.

Auch der vorgestellte Ansatz zur Relaxierung von Zeitbedingungen (siehe Abschnitt 4.2.6) hilft an dieser Stelle nur bedingt weiter, da das berechnete minimale Netz der Aktivität Weitere Vorbereitungen eine minimale Dauer von null und eine maximale Dauer von einem Tag zuweisen würde, diese aber nicht mit dem maximalen Abstand von 3 Stunden zwischen den Aktivitäten Weitere Vorbereitungen und Untersuchung durchführen in Einklang zu bringen ist (siehe Beispiel 5.2). An diesem Beispiel erkennt man gut eine Besonderheit der Definition des Schedules bzw. des freien Schedules: Damit der Agent lediglich auf die lokalen Zeitbedingungen seiner Aktivität achten muss, darf die Differenz zwischen der minimalen und der maximalen Dauer der Aktivität nicht größer als die Spanne des Zeitabstandes zwischen der Aktivität und ihrem Nachfolger sein. Ansonsten ist es nicht möglich, die Zeit-Schwankungen innerhalb der Spezifikation aufzufangen und es kommt zu Zeitfehlern. Beispiel 5.2 verdeutlicht nochmals den Hintergrund dieser Einschränkung. Allgemein muss zwischen zwei Aktivitäten folgende, mit in die Bedingungen für den Lösungsraum der freien Schedules eingeflossene, Bedingung gelten:

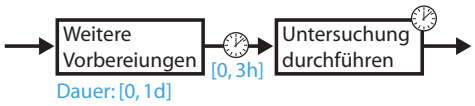
$$\begin{array}{rrcl} & D_i^{max} + \min(c_{i_{end}j_{start}}) & \leq & D_i^{min} + \max(c_{i_{end}j_{start}}) \\ \Leftrightarrow & D_i^{max} - D_i^{min} & \leq & \max(c_{i_{end}j_{start}}) - \min(c_{i_{end}j_{start}}) \end{array} \tag{5.1}$$

Diese Bedingung besagt, dass die maximale Dauer der Aktivität $i$ zusammen mit dem minimalen Abstand zwischen den Aktivitäten $i$ und $j$ kleiner oder gleich der minimalen Dauer der Aktivität $i$ zusammen mit dem maximalen Abstand zwischen $i$ und $j$ sein muss. Anders ausgedrückt bedeutet dies, dass der Abstand zwischen $i$ und $j$ in der Lage sein muss, die Schwankungen der Dauer von Aktivität $i$ aufzufangen.

Es müssen also zumindest für einige der Aktivitäten eines Prozessschemas entsprechende Maximal-Dauern und eventuell auch Minimal-Dauern angegeben werden. Um den größten Nutzen aus der Zeitrechnung ziehen zu können, sollte dies sinnvollerweise für alle Aktivitäten des Prozessschemas geschehen. Wie in Abschnitt 2.2.2.1 diskutiert gibt es verschiedene Quellen, um an diese Werte zu gelangen. Exemplarische Werte für den Beispiel-Prozess und der daraus resultierenden Startzeitintervalle der freien Schedules sind in Beispiel 5.3 zu finden.

Wie man an den dort dargestellten Startzeitintervallen der freien Schedules erkennt, sind die spätesten Startzeitpunkte für die Aktivitäten bisher wenig aussagekräftig. Dies liegt vor allem daran, dass das Prozessschema bisher kaum Maximal-Abstände enthält, welche die spätesten Startzeitpunkte begrenzen würden. Um die Aussagekraft der Zeitdaten zu erhöhen, ist es daher sinnvoll, einige angemessene Maximal-Abstände und/oder eine maximale Gesamtdauer (ein Maximal-Abstand zwischen Start und Ende des Prozessschemas) für den Prozess anzugeben.

Wird beim Prozess aus Beispiel 5.1 eine Maximal-Dauer von einem Tag für die Aktivität Weitere Vorbereitungen angesetzt und der Termin der Aktivität Untersuchung durchführen als fest betrachtet, so kann aufgrund des Maximal-Abstandes von 3 Stunden die Einhaltung der Zeitbedingung zwischen den beiden Aktivitäten nicht mehr sichergestellt werden.




Um garantieren zu können, dass der für Untersuchung durchführen zuständige Agent diese Aktivität zum vorgesehenen Zeitpunkt durchführen kann, muss der Startzeitpunkt für die Aktivität Weitere Vorbereitungen einen Tag früher liegen, damit der Agent auch die angegebene Zeit zu Verfügung hat. Falls der zuständige Agent für die weiteren Vorbereitungen jedoch nur eine Stunde benötigt, so liegt zwischen dem Ende der Aktivität Weitere Vorbereitungen und dem Start der Aktivität Untersuchung durchführen ein zeitlicher Abstand von 23 Stunden. Dieser widerspricht jedoch dem spezifizierten maximalen Abstand von drei Stunden zwischen den beiden Aktivitäten.
Um das Problem zu lösen, gibt es in diesem Fall verschiedene Möglichkeiten:

- Die minimale Dauer der Aktivität Weitere Vorbereitungen kann erhöht werden, z. B. auf 21 Stunden.
- Die maximale Dauer der Aktivität Weitere Vorbereitungen kann begrenzt werden, z. B. auf 3 Stunden.
- Der maximale Abstand zwischen den beiden Aktivitäten kann erhöht werden, z. B. auf einen Tag.

Beispiel 5.2: Einschränkungen für Dauer und Abstand

## 5.3 Schlüsselaktivitäten

Die Festlegung des Termins der Aktivität Untersuchung durchführen kann auf verschiedene Arten geschehen. Zum einen ist es möglich, dass die Aktivität Termin vereinbaren eine externe Terminverwaltung startet und der Termin dort hinterlegt wird, zum anderen könnte die Aktivität den Zeitpunkt des Termins in ein Datenelement schreiben. Je nach Variante wird bei der Aktivität Untersuchung durchführen eine entsprechende Funktion für die Schlüsselaktivität hinterlegt, die bei der Aktualisierung des Zeitmodells angewendet wird, um den entsprechenden Termin zu erhalten.

Betrachtet man das Zeitmodell des Prozessschemas beziehungsweise dessen minimales Netz (siehe Anhang B), sieht man weiter, dass der Termin der Aktivität Untersuchung

**Aktivitäten Dauern und freie Schedules**

| Aktivität | Dauer | Startzeitintervall |
|---|---|---|
| StartWF | $[0min, 0min]$ | $[0min, 0min]$ |
| Maßnahme anordnen | $[10min, 1h]$ | $[0min, \infty]$ |
| Termin vereinbaren | $[10min, 15min]$ | $[1h, \infty]$ |
| AND_SPLIT | $[0min, 0min]$ | $[1h15min, \infty]$ |
| Patient aufklären | $[30min, 2h]$ | $[1h15min, \infty]$ |
| Patient vorbereiten | $[10min, 2h]$ | $[1h15min, \infty]$ |
| AND_JOIN | $[0min, 0min]$ | $[3h15min, \infty]$ |
| Weitere Vorbereitungen | $[20min, 1h]$ | $[23h55min, \infty]$ |
| Untersuchung durchführen | $[1h, 2h]$ | $[1d3h15min, \infty]$ |
| XOR_SPLIT | $[0min, 0min]$ | $[1d5h15min, \infty]$ |
| Kurzbefund erstellen | $[15min, 30min]$ | $[1d5h15min, \infty]$ |
| Kurzbefund lesen | $[5min, 1h]$ | $[1d5h45min, \infty]$ |
| XOR_JOIN | $[0min, 0min]$ | $[1d6h45min, \infty]$ |
| Befund erstellen | $[30min, 5h]$ | $[1d6h45min, \infty]$ |
| Befund lesen | $[15min, 1h]$ | $[1d11h45min, \infty]$ |
| EndWF | $[0min, 0min]$ | $[1d12h45min, \infty]$ |

Spezial-Knoten wie START_WF, AND_SPLIT, AND_JOIN etc. werden in diesem Beispiel vom System ausgeführt und besitzen daher keine nennenswerte Zeitdauer (d. h. $[0min, 0min]$). Dies ist jedoch nicht zwangsläufig so, beispielsweise kann eine XOR_SPLIT eine Benutzerabfrage beinhalten, die eine gewisse Zeit benötigt. Da das Zeitmodell jedoch nicht zwischen normalen Aktivitäten und Spezial-Knoten unterscheide, müssen diese beiden Fälle auch nicht getrennt betrachtet werden.

Beispiel 5.3: Zeitdauern für den Prozess

durchführen zunächst keine Rolle spielt. Dies liegt vor allem daran, dass er bisher noch unbekannt ist und daher lediglich durch ein Constraint mit dem Wertebereich $[0, \infty]$ repräsentiert werden kann. Jedoch wird der mögliche Wertebereich für den Termin, durch die verschiedenen Abstandsbedingungen, relativ zum Start des Prozesses eingeschränkt, d. h. der ursprüngliche Wertebereich $[0, \infty]$ des entsprechenden Constraints wird bei Ermittlung des minimalen Netzes reduziert. Im konkreten Beispiel bedeutet dies, dass der Termin frühestens etwa 1 Tag und 4 Stunden nach dem Start des Prozesses stattfinden darf. Wird eine Maximal-Dauer oder ein Maximal-Abstand hinzugefügt, so kann es zusätzlich sein, dass der Termin einen spätesten Zeitpunkt zugewiesen bekommt, zu dem er stattfinden kann. Diese Bedingungen müssen bei der Vergabe des Termins in der Aktivität Termin vereinbaren beachtet werden. Das heißt auch, die Aktivität Termin vereinbaren muss vom Prozess-Management-System beziehungsweise dessen Zeitmanagement mitgeteilt bekommen, in welchem zeitlichen Rahmen der Termin festgelegt werden kann.

Die Aktivität ist dann dafür verantwortlich darauf zu achten, dass der Benutzer diese Vorgabe einhält. Geschieht dies nicht oder ignoriert der Benutzer die Vorgabe, kommt es unweigerlich zu einem Zeitfehler und damit zu einer Eskalation, sobald das Zeitmodell mit dem neu festgelegten Termin aktualisiert wird.

## 5.4 Fristen

Eine andere interessante Anwendung von Terminen beziehungsweise Fristen ist die Festlegung der Gültigkeitsdauer eines Prozesses. Dies ist zum Beispiel dann relevant, wenn ein Prozess aufgrund betrieblicher Änderungen nur noch bis zu einem gewissen Datum gültig ist und/oder der ablösende Prozess erst ab einem anderen Datum gestartet werden darf. Vor allem im ersten Fall bietet es sich an, der End-Aktivität des Prozessschemas einen festgelegten spätesten Startzeitpunkt zu geben. Das Zeitmanagement berechnet aus diesem dann den spätesten Termin, bis zu dem dieses Prozessschema noch gestartet werden kann, sodass alle seine Instanzen innerhalb der Gültigkeitsdauer abgeschlossen werden können. Versucht ein Benutzer dennoch nach diesem Termin eine neue Prozessinstanz zu erzeugen, so kommt es unmittelbar nach dem Erzeugen zu Eskalation. In diesem Fall kann der Prozess entweder abgebrochen und das ablösende Prozessschema gestartet werden, oder das Prozessschema kann so angepasst werden, dass es sämtliche Zeitbedingungen erfüllt.

Eine Bedingung der Art „nur von 8 bis 16 Uhr“, wie bei der Aktivität Patient vorbereiten, kann ebenfalls durch eine Schlüsselaktivität modelliert werden. Dazu wird bei der entsprechenden Aktivität eine Schlüsselaktivitäts-Funktion hinterlegt, welche bei der Übergabe der aktuellen Ausführungsintervalle keinen festen Termin liefert, sondern die Intervalle so einschränkt, dass der früheste Startzeitpunkt frühestens um 8 Uhr und der späteste Endzeitpunkt spätestens um 16 Uhr ist.

## 5.5 Fazit

Wie gezeigt wurde, gibt es bei der Zeitmodellierung eines Prozesses einiges zu beachten, weswegen die entsprechend zu modellierenden Zeitbedingungen gut durchdacht sein sollten. Nachdem der Prozess modelliert wurde und auch die Konsistenz der Zeitbedingungen sichergestellt ist, steht einer Ausführung des Prozesses aus Sicht des Zeitmanagements nichts mehr im Wege. Welche Abläufe bei der Ausführung einer Prozessinstanz stattfinden und welche Besonderheiten es dabei zu beachten gibt, wird im nächsten Abschnitt diskutiert.

# 6 Interpretation zeitlicher Aspekte zur Laufzeit

Wird eine neue Prozessinstanz erzeugt, wird vom zugrunde liegenden Prozessschema zunächst eine logische Kopie angelegt und diese um die nötigen Laufzeitinformationen (siehe Abschnitte 2.1.4 und 4.1.4) angereichert. Dieser Vorgang wird auch als Instanzierungsphase des Prozessschemas bezeichnet. Im Fall des Zeitmanagements müssen in dieser Phase einige weitere Aufgaben durchgeführt werden, die in Abschnitt 6.1 erläutert werden.

Im Anschluss daran werden die Aufgaben während der Ausführung eines Prozesses diskutiert (Abschnitt 6.2). Hier muss das Zeitmodell der Prozessinstanz zu bestimmten Zeitpunkten aktualisiert sowie die Einhaltung der Zeitbedingungen überwacht werden. Bei Verletzung einer Zeitbedingung muss ein Zeitfehler ausgelöst werden, der meist zu einer Eskalation führt, um die Konsistenz der Prozessinstanz wiederherzustellen oder sie abzubrechen. Daneben werden verschiedene Möglichkeiten zur Optimierung des in Kapitel 4 vorgestellten Zeitplanungs-Algorithmus vorgestellt, die während dieser Phase verwendet werden können.

Auch bei sorgfältiger Planung der Zeitbedingungen eines Prozesses kann es zu Zeitfehlern kommen. Wie das Prozess-Management-System hierauf reagieren muss und welche Hilfestellungen es dem Benutzer bei der Behebung des Zeitfehlers geben kann, wird in Abschnitt 6.3 erläutert. Dabei werden auf Basis der vorliegenden Ausführungszustand-Zeitzustand-Kombination verschiedene Arten von Zeitfehlern unterschieden und entsprechend differenziert behandelt.

Eine Betrachtung des zugrunde liegenden Laufzeitsystems (Abschnitt 6.4) rundet die vorhergehenden Abschnitte ab. Hier wird diskutiert, wie das Prozess-Management-System aufgebaut sein muss, um die während der Instanzierung und der Ausführung anfallenden Aufgaben möglichst gut unterstützen zu können.

Abschnitt 6.5 befasst sich mit dynamischen Modifikationen an einer Prozessinstanz während deren Ausführung und welchen Einfluss das Zeitmanagement hierauf hat. Dabei werden zwei Arten von Modifikationen, namentlich status- und strukturändernde Operationen, unterschieden und entsprechend getrennt diskutiert.

Eine wichtige Anwendung des Zeitmanagements zur Laufzeit ist die Prognose von Zeitdaten, welche in Abschnitt 6.6 diskutiert wird. Dabei wird auch auf verschiedene Anwendungsmöglichkeiten der prognostizierten Zeitdaten (z. B. ein Ressourcenmanagement) eingegangen.

Abgeschlossen wird das Kapitel durch eine Zusammenfassung in Abschnitt 6.7, welche nochmals kurz auf die wichtigsten Punkte eingeht.

## 6.1 Instanzierungsphase

Während der Instanzierungsphase wird zunächst eine logische Kopie des *Prozessschemas* erstellt, die man als *Prozessinstanz* bezeichnet. Dies geschieht sowohl, damit die Prozessinstanz um Laufzeitinformationen (z. B. aktuelle Ausführungszustände der Aktivitäten) angereichert werden kann als auch, damit sich während der Laufzeit durchgeführte Veränderungen an einer Prozessinstanz (z. B. das Setzen von Zeitpunkten) nicht auf die anderen Instanzen desselben Prozessschemas auswirken.

Als Nächstes setzt das Zeitmanagement das Startzeitintervall der Start-Aktivität der Prozessinstanz auf den aktuellen Zeitpunkt. Danach muss das Zeitmodell der Prozessinstanz mit dem neuen Zeitdatum aktualisiert und das minimale Netz sowie die angepasste Lösungsmenge der freien Schedules berechnet werden. Hiermit lassen sich dann auch erste Zeitpunkte der anderen Aktivitäten abschätzen.

Besitzt der Prozess bzw. das Prozessschema, wie in Kapitel 5 diskutiert, eine Gültigkeitsdauer, kann es an dieser Stelle bereits zu einer Inkonsistenz des Zeitmodells kommen. In diesem Fall kann der Prozess noch nicht oder nicht mehr gestartet werden. Daraufhin muss der Benutzer über dieses Problem informiert und die Prozessinstanz entweder abgebrochen oder vom Benutzer so an die Gegebenheiten angepasst werden, dass die Instanz wieder ein konsistentes Zeitmodell besitzt. Ein Beispiel für eine solche Anpassung ist die Aufhebung der zunächst vorgegebenen Gültigkeitsdauer für diese spezielle Prozessinstanz oder verschiedene andere Möglichkeiten zur Behandlung von Zeitfehlern.

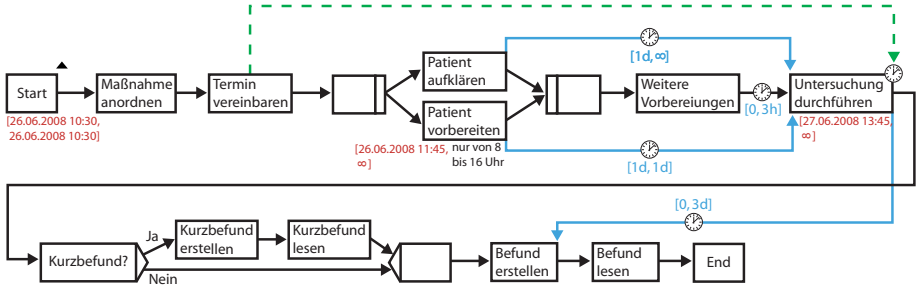
Sofern eine der externen Komponenten in der Lage ist, eine der dynamischen Zeitbedingungen aus dem Startzeitpunkt der Prozessinstanz abzuleiten, können an dieser Stelle die entsprechenden dynamischen Dauern und Abstände sowie Fristen aktualisiert werden. Dies bietet sich besonders bei Fristen der Art „jeder erste Tag des Monats“ oder ähnlich strukturierten Zeitbedingungen an. Auch hier ist eine Neuberechnung des Zeitmodells und der freien Schedules nötig.

Damit ist die Instanzierungsphase aus Sicht des Zeitmanagements abgeschlossen und die Prozessinstanz kann dem Prozess-Management-System zur Ausführung übergeben werden. Die Instanzierungsphase einer Prozessinstanz ist in Beispiel 6.1 zu finden.

## 6.2 Ausführungsphase

Während der Ausführungsphase ist die Dauer einzelner Aktivitäten unbestimmt. Dies kann die Konsistenz von Prozessinstanzen beeinflussen, deren zugrunde liegende Prozessschemata während der Modellierungs- und Instanzierungsphase konsistent waren. Daher muss das Zeitmodell einer Prozessinstanz an verschiedenen *Kontrollpunkten* aktualisiert

Die Graphik zeigt eine Prozessinstanz des Prozesses aus Beispiel 5.1 mit den Zeitbedingungen aus Beispiel 5.3. Die Instanz wurde am 26.06.2008 um 10:30 gestartet. Wie ersichtlich ist, ergeben sich aus dem Startzeitpunkt der Prozessinstanz bereits mögliche Zeitpunkte für einige der anderen Aktivitäten, die hier am Beispiel der beiden Aktivitäten Patient vorbereiten und Untersuchung durchführen exemplarisch dargestellt sind.




Beispiel 6.1: Instanzierte Prozessinstanz

und erneut auf seine Konsistenz überprüft werden. Die Kontrollpunkte dienen dabei dazu, festzulegen, an welchen Punkten der Ausführung einer Prozessinstanz eine Aktualisierung des Zeitmodells zu erfolgen hat. Zwischen den Aktualisierungen des Zeitmodells müssen die Ausführungszeiten der laufenden Aktivitäten überwacht werden, um beim Auftreten von Zeitfehlern eine entsprechende Eskalation auslösen zu können.

Wird eine neue Aktivität gestartet, d. h. wechselt ihr Ausführungszustand von ACTIVATED nach STARTED, steht ihr Startzeitpunkt fest. Dementsprechend wird zunächst ihr Startzeitintervall der Prozessinstanz durch den aktuellen Zeitpunkt ersetzt. Daraufhin müssen die Zeitdaten der Prozessinstanz, d. h. das Zeitmodell und die Menge der freien Schedules, durch die entsprechenden Algorithmen aktualisiert werden. Dazu werden sie erneut mit dem Zeitmodell der aktualisierten Prozessinstanz ausgeführt, um so die neuen Zeitdaten zu erhalten.

In einem weiteren Schritt wird überprüft, ob angesichts der Änderungen an den Zeitdaten dynamische Zeitbedingungen aktualisiert bzw. ermittelt werden können. Sobald die aktuellen Zeitdaten feststehen, müssen die Zeitzustände der laufenden Aktivitäten kontinuierlich überwachen und Änderungen der Zustände gemeldet werden. Die Bedingungen für die verschiedenen Zeitzustände, in Abhängigkeit des aktuellen Ausführungszustandes der Aktivität, wurden in Abschnitt 4.1.4 diskutiert und sind in verkürzter Form nochmals Tabelle 6.1 zu entnehmen.

Nach Ende der Aktivität, d. h., sobald ihr Ausführungszustand nach COMPLETED wechselt, muss das Zeitmodell der Prozessinstanz erneut aktualisiert und der neue Zeitzustand an die Ausführungs-Komponente gemeldet werden. Diesmal wird sowohl die Dauer der Aktivität als auch ihr Endzeitintervall durch die entsprechenden Werte ersetzt.

| | EARLY | ON_TIME | LATE |
|---|---|---|---|
| NOT_ACTIVATED | — | — | $SZ_i^{start} < \mathcal{NOW}$ |
| ACTIVATED | $\mathcal{NOW} < FZ_i^{start}$ | $FZ_i^{start} \leq \mathcal{NOW} \leq SZ_i^{start}$ | $SZ_i^{start} < \mathcal{NOW}$ |
| STARTED / SUSPENDED | $\mathcal{NOW} - FZ_i^{start} < D_i^{min}$ | $D_i^{min} \leq \mathcal{NOW} - FZ_i^{start} \leq D_i^{max}$ | $D_i^{max} < \mathcal{NOW} - FZ_i^{start}$ |
| COMPLETED | $\mathcal{NOW} < FZ_i^{end}$ | $FZ_i^{end} \leq \mathcal{NOW} \leq SZ_i^{end}$ | $SZ_i^{end} < \mathcal{NOW}$ |

Die übrigen Ausführungszustände (FAILED und SKIPPED) besitzen immer den Zeitzustand UNDEFINED.

Tabelle 6.1: Zeit- und Ausführungszustände (siehe Abschnitt 4.1.4)

Daraufhin findet erneut eine Aktualisierung der Zeitdaten (vor allem der Menge der freien Schedules) der Prozessinstanz und – falls nötig – der dynamischen Zeitbedingungen statt.

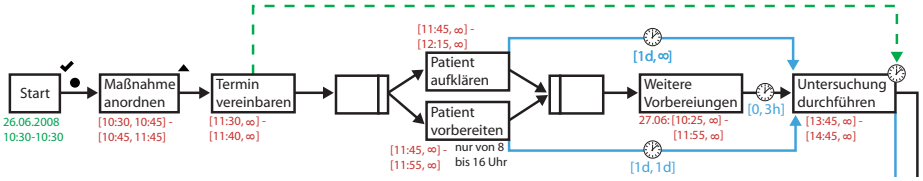
Als Nächstes werden die minimalen Abstände und Startzeitintervalle der nachfolgenden Aktivitäten ermittelt und diese beim Prozess-Management-System solange für die Ausführung gesperrt bzw. in der Arbeitsliste als noch nicht ausführbar markiert, bis die entsprechende minimale Zeitdauer vergangen ist. Während dieser Phase besitzen die Nachfolger-Aktivitäten die Ausführungszustand-Zeitzustand-Kombination ACTIVATED-EARLY.

Meldet das Zeitmanagement für eine Aktivität den Zeitzustand LATE oder die Ausführungszustand-Zeitzustand-Kombination COMPLETED-EARLY, ist das Prozess-Management-System gezwungen, hierauf zu reagieren, indem es eine Eskalation auslöst. Daraufhin muss die für die Eskalation zuständige Komponente mithilfe des Benutzers die Prozessinstanz bzw. deren Zeitmodell so modifizieren, dass für diese Aktivität wieder eine erlaubte Ausführungszustand-Zeitzustand-Kombination gilt (siehe Abschnitt 6.3). Zugleich muss natürlich darauf geachtet werden, dass der übrige Teil der Prozessinstanz weiterhin zeitkonsistent ist. Ein beispielhafter Ablauf der Ausführung einer Prozessinstanz ist in Beispiel 6.2 dargestellt.

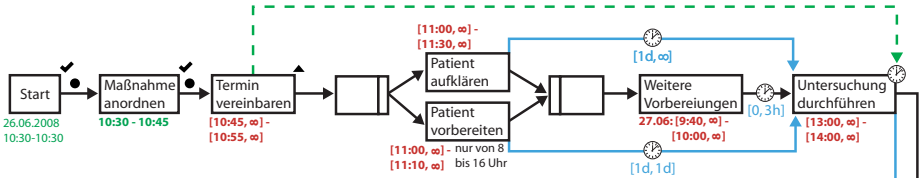
### 6.2.1 Optimierung der Algorithmen zur Laufzeit

Findet während der Instanzierungs- oder Ausführungsphase eine Veränderung an den Zeitdaten der Prozessinstanz statt, muss meist nicht das komplette Zeitmodell neu berechnet werden. Häufig genügt es, die veränderten Constraints des zuvor berechneten minimalen Netzes zu aktualisieren und aus diesem dann erneut mittels dem entsprechenden Algorithmus ein aktualisiertes minimales Netz zu berechnen (siehe Beispiel 6.3).
Als weitere Optimierung kann in diesem Fall beispielsweise bei Algorithmus 1 die initiale Menge der zu untersuchenden Pfade (Zeile 2) auf die mit den veränderten Constraints in Beziehung stehenden Pfade (d. h. Pfade der Länge 3 von denen das veränderte Constraint ein Teilstück ist) reduziert werden. Für den Fall, dass beispielsweise nur die Constraints

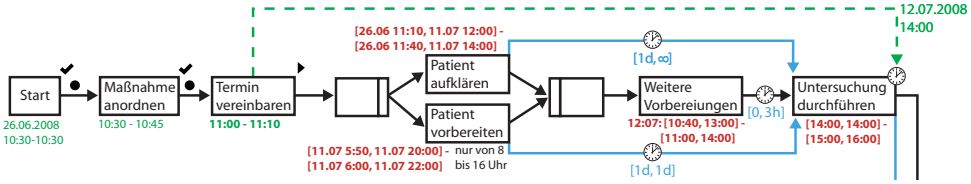
Die Prozessinstanz wurde am 26.06.2008 um 10:30 gestartet. Aus dem Startzeitpunkt lassen sich weitere Zeitdaten ermitteln[a], etwa das Startzeitintervall der Aktivität Maßnahme anordnen.




Nach dem Ausführen der Aktivität Maßnahme anordnen wird das Zeitmodell aktualisiert, womit sich ein neuer frühester Startzeitpunkt für die Aktivität Termin vereinbaren und einige andere Aktivitäten ergeben.



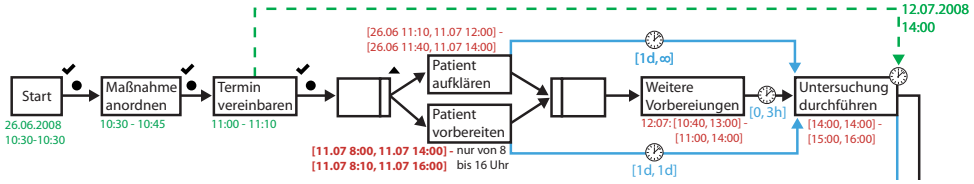

Durch die Aktivität Termin vereinbaren wird der Termin für die Aktivität Untersuchung durchführen auf den 12.07.08 14:00 Uhr festgelegt. Sowohl durch den Ausführungszeitpunkt der Aktivität Termin vereinbaren als auch durch das Festlegen des Termins, ergeben sich Änderungen an den Start- und Endzeitintervallen. So werden beispielsweise die beiden Intervalle für die Aktivität Weitere Vorbereitungen auf wenige Stunden eingeschränkt.




[a] Zeitdaten in Rot stellen Prognosen und Zeitdaten in Grün stellen durch die Ausführung festgelegte Werte dar.

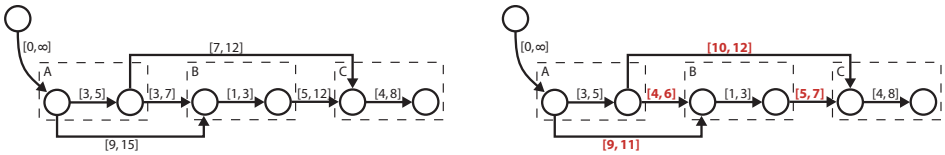
Beispiel 6.2: Ausführung einer Prozessinstanz

Durch das Festlegen des Termins der Aktivität Untersuchung durchführen, schränken sich die Intervalle der Aktivität Patient vorbereiten derart ein, dass die entsprechende Funktion in der Lage ist, die dynamische Zeitbedingung „nur von 8 bis 16 Uhr“ für die Zeitpunkte der Aktivität Patient vorbereiten umzusetzen. Damit ergeben sich weitere Einschränkungen für die Start- und Endzeitintervalle der Aktivität Patient vorbereiten, die sich in diesem Fall jedoch nicht auf andere Aktivitäten auswirken.

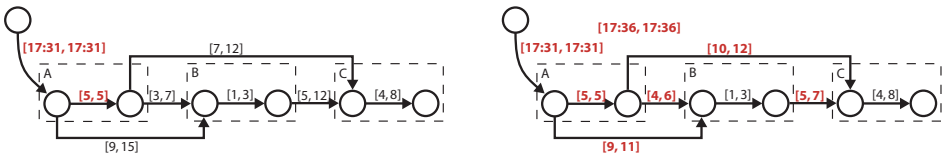



Beispiel 6.2: Ausführung einer Prozessinstanz (forts.)

Die Abbildung zeigt das Zeitmodell eines Prozessschemas (links) zusammen mit dessen minimalem Netz (rechts). Bei der Berechnung des minimalen Netzes wurden einige der Constraints eingeschränkt (rot markiert).




Wird die Aktivität A um 17:31 Uhr gestartet und dauert 5 Minuten so wird das Zeitmodell wie auf der linken Seite dargestellt aktualisiert. Aus diesem ergibt sich das auf der rechten Seite dargestellte minimale Netz. Dasselbe Ergebnis erreicht man auch, indem man das oben rechts dargestellte minimale Netz entsprechend aktualisiert.




Beispiel 6.3: Optimierte Aktualisierung eines Zeitmodells zur Laufzeit

in $C \subseteq \mathbf{\Gamma}$ verändert wurden, ergibt sich als initiale Pfadmenge in Zeile 2 von Algorithmus 1 folgende Menge:

$$Q = \bigcup_{c_{ij} \in C} \{(i,j,k),(k,i,j) | (0 \leq k \leq n) \wedge (i \neq k \neq j)\}.$$

Die einzige Bedingung für die Anwendbarkeit dieser Regel ist, dass die veränderten Constraints eingeschränkt wurden. Ein Constraint $c'_{ij}$ ist gegenüber einem Constraint $c_{ij}$ *eingeschränkt*, falls $c'_{ij} \subseteq c_{ij}$ gilt. Wurde eines der veränderten Constraints auch erweitert, d. h., es gilt nicht: $c'_{ij} \subseteq c_{ij}$, so muss das Zeitmodell aktualisiert und daraus ein neues minimales Netz berechnet werden. In diesem Fall kann auch die initiale Pfadmenge nicht reduziert werden.

Diese Regel ist möglich, da der Algorithmus zur Zeitrechnung (Algorithmus 1) selbst auf der Einschränkung von Constraints beruht. Daher spielt es keine Rolle, ob ein Constraint von einer vorherigen Iteration der While-Schleife des Algorithmus oder bereits vorab eingeschränkt wurde. Die initiale Pfadmenge kann in diesem Fall ebenfalls reduziert werden, da von den gemachten Einschränkungen zunächst nur die Pfade der reduzierten Pfadmenge betroffen sind (siehe auch Zeile 8 von Algorithmus 1). Daher reicht es aus, nur diese zu betrachten. Ergeben sich daraus weitere betroffene Pfade, werden diese automatisch vom Algorithmus ermittelt und der Pfadmenge $Q$ hinzugefügt (Algorithmus 1 Zeile 8).

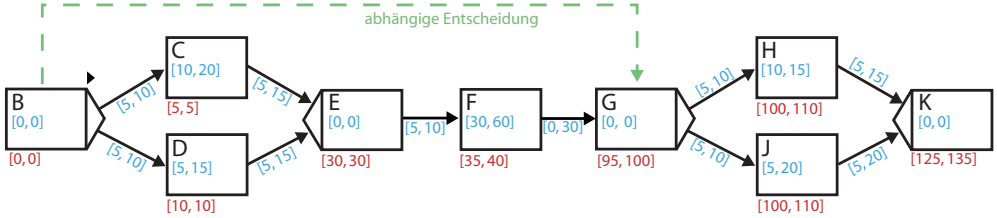
Wurde ein Constraint des Zeitmodells auch erweitert (d. h. $c'_{ij} \not\subseteq c_{ij}$), so lässt sich diese Optimierung nicht mehr anwenden. Es ist dann nötig, das minimale Netz aus dem ursprünglichen Zeitmodell neu zu berechnen. Durch eine Graphanalyse ist es möglich, die durch die Erweiterung des Intervalls betroffenen Bereiche zu ermitteln und nur diese neu zu berechnen. Jedoch hebt sich der dadurch gewonnene Vorteil bei der Aktualisierung des minimalen Netzes durch den nötigen Aufwand zur Graphanalyse wieder auf, weswegen dies nicht weiter verfolgt wird.

Die eben diskutierten Möglichkeiten zur Optimierung des Konsistenz-Algorithmus können auch im Rahmen der Ermittlung der freien Schedules verwendet werden. Wurden die in den Bedingungen für die Menge der freien Schedules (siehe Abschnitt 4.2.4.2 Seite 90) verwendeten Constraints für ein Aktivitäten-Paar durch die Aktualisierung nicht verändert, ändert sich auch das Zeitmodell für den Lösungsraum an dieser Stelle nicht. Dadurch kann, wie oben diskutiert, auf das bereits berechnete minimale Netz zurückgegriffen und nur dieses aktualisiert werden. Ebenso kann für die Anwendung von Algorithmus 1 eine reduzierte initiale Pfadmenge verwendet werden.

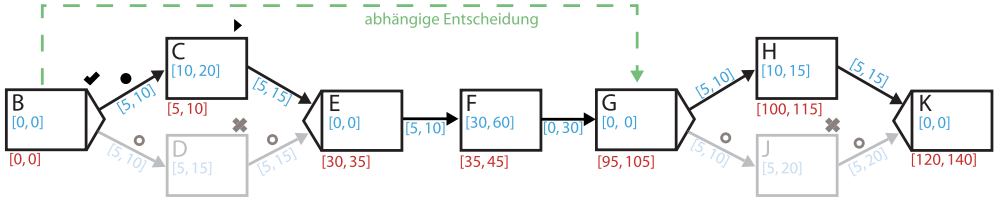
Das Löschen von Instanztypen (d. h. von Pfaden von Alternativ-Verzweigungen) bzw. das Ignorieren dieser bei der Transformation in das Zeitmodell, ist eine weitere Möglichkeit einer Laufzeit-Optimierung. Dies ist möglich, sobald feststeht, dass diese Instanztypen nicht mehr zur Ausführung kommen können. Hierbei können vor allem auch bekannte Abhängigkeiten zwischen verschiedenen Alternativ-Verzweigungen ausgenutzt werden

(siehe Beispiel 6.4). Ähnliches gilt auch für Schleifen, gesetzt den Fall, dass hier von den Algorithmen mehrere mögliche Iterationen zugleich betrachtet werden (siehe Abschnitt 4.2.5). Durch diese Optimierung lässt sich nicht nur der nötige Rechenaufwand zur Aktualisierung des Zeitmodells und der freien Schedules reduzieren, sondern auch die Qualität der Zeit-Prognose verbessern, da die Intervalle nach der Verzweigung nicht mehr verschiedene Instanztypen zugleich repräsentieren.

Das Bild stellt einen Prozess dar, bei dem bekannt ist, dass die Entscheidung des zweiten XOR-Blocks die gleiche ist, wie beim ersten XOR-Block. Das heißt fällt die Entscheidung des XOR-Blockes $B \mapsto E$ auf den oberen Pfad (Aktivität $C$), so fällt auch die Entscheidung des XOR-Blockes $G \mapsto K$ auf den oberen Pfad (Aktivität $H$) und umgekehrt.




Damit können, wenn bei der Ausführung des XOR-Split-Knoten $B$ die Entscheidung auf den oberen Pfad fällt, sowohl der untere Pfad des Blocks $B \mapsto E$ als auch der untere Pfad des Blocks $G \mapsto K$ aus der Prozessinstanz beziehungsweise deren Zeitmodell entfernt werden. Dadurch verändern sich, wie man an den beiden Abbildungen erkennen kann, auch einige der vorhergesagten Zeiten, da jetzt genauere Aussagen getroffen werden können.




Beispielsweise ändert sich, nachdem die XOR-Entscheidung gefallen ist, das Startzeitintervall der Aktivität $F$ von $[35, 40]$ auf $[35, 45]$ und das der Aktivität $K$ von $[125, 135]$ auf $[120, 140]$. Dies entspricht jeweils den Werten, die man erhält, wenn man nur diesen einen Instanztyp betrachtet.

Beispiel 6.4: Ausnutzung abhängiger Alternativ-Entscheidungen

### 6.2.2 Aktualisierung dynamischer Zeitbedingungen

Nachdem das Zeitmodell der Prozessinstanz an einem Kontrollpunkt auf den neuesten Stand gebracht worden ist, gilt es zu überprüfen, ob sich hierdurch Änderungen für die dynamischen Zeitbedingungen ergeben. Hierzu werden für jede dynamische Zeitbedingung, deren Zeitbedingung im Zeitmodell sich verändert hat, erneut die entsprechende Funktion auf den aktuellen Wert der Zeitbedingung angewendet. Entspricht das Ergebnis der Funktion nicht ihrer Eingabe, d. h. gilt $f([x]) \subsetneq [x]$, wird das erhaltene Intervall als neuer Wert für die Zeitbedingung in das Zeitmodell eingetragen. Nachdem auf diese Weise alle dynamischen Zeitbedingungen überprüft worden sind, wird, falls eine Zeitbedingung aktualisiert wurde, der Zeitkonsistenz-Algorithmus erneut gestartet, um die daraus resultierenden Einschränkungen zu erhalten. Dabei kann es zu weiteren Einschränkungen an den dynamischen Zeitbedingungen kommen, was einen weiteren Durchlauf der eben beschriebenen Funktion nötig macht. Ein Beispiel für dieses Vorgehen ist im letzten Teil von Beispiel 6.2 zu finden. Außerdem ist der gesamte Ablauf nochmals in Algorithmus 4 zusammengefasst. Dabei stellt die Funktion `propagate` die Anwendung von Algorithmus 1 auf das gegebene Zeitmodell dar.

Der Algorithmus terminiert, da in jeder Iteration mindestens eines der Intervalle des Zeitmodells weiter eingeschränkt wird. Daher findet der Algorithmus entweder ein Zeitmodell, dass nicht mehr weiter eingeschränkt wird, oder er bricht ab, falls das resultierende Zeitmodell inkonsistent ist. In diesem Fall muss ein Zeitfehler ausgelöst werden, wobei auf die in Abschnitt 4.2.4.1 vorgestellte Vorgehensweise zur Lokalisierung der inkonsistenten Stelle zurückgegriffen werden kann.

Es ist auch möglich, dass die Funktion für eine dynamische Zeitbedingung, eine Aktualisierung des Zeitmodells anfordert. Dies ist beispielsweise dann der Fall, wenn der Wert der Funktion von einem Datenelement zur Verfügung gestellt wird und dieses durch eine Aktivität verändert wurde. Eine solche Anforderung stellt für das Zeitmanagement zunächst lediglich einen Hinweis dar, dass eine Aktualisierung sinnvoll wäre. Das Zeitmanagement3 kann die Aktualisierung jedoch so lange aufschieben, bis der konkrete Wert des Zeitdatums benötigt wird, die entsprechende Aktivität bzw. die entsprechende Zeitkante also aktiviert werden soll. Je länger die Aktualisierung jedoch hinauszögert wird, desto größer ist das Risiko, das die Funktion aufgrund der gegebenen Bedingungen keine passenden Werte mehr findet und ein leeres Intervall zum Ergebnis hat.

Die Aktualisierung der dynamischen Zeitbedingungen spielt auch bei der im nächsten Abschnitt diskutierten Selektion von Kontrollpunkten eine Rolle. Egal welche Strategie für die Selektion der Kontrollpunkte gewählt wird, muss vor der „Aktivierung“ einer *Schlüsselaktivität* oder einer dynamischen Zeitkante immer ein Kontrollpunkt eingefügt werden, um den aktuellsten Wert für das entsprechende Zeitdatum zu erhalten.

**Input** : Aktualisiertes minimales Netz eines Zeitgraph $M = (\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$ mit $\mathbf{\Gamma} = \{c_{ij} | 0 \leq i, j \leq n\}$; Menge der veränderten Constraints $C \subseteq \mathbf{\Gamma}$
**Output** : Aktualisiertes minimales Netz oder inkonsistent
**Funktionen** :
`getDynamicTimeSpanFunction`$(i, j)$ bestimmt die Funktion zur Bestimmung des neuen Wertes des Constraint $i \rightarrow j$
`getDynamicTimeLimitFunction`$(a)$ bestimmt die Funktion zur Bestimmung des Termin der Aktivität $a$
`getActivityStartTime`$(a)$ bestimmt das Constraint für die Startzeit der Aktivität $a$
`getActivityEndTime`$(a)$ bestimmt das Constraint für die Endzeit der Aktivität $a$

```
1  begin
2      while C ≠ ∅ do
3          foreach c_ij ∈ C do
4              g = getDynamicTimeSpanFunction(i, j);
5              if g ≠ ∅ then
6                  c'_ij = g(c_ij);
7                  if c'_ij ⊂ c_ij then c_ij = c'_ij;
8              end
9          end
10         foreach a ∈ activitiesOf(C) do
11             f = getDynamicTimeLimitFunction(a);
12             if f ≠ ∅ then
13                 c_0a_start = getActivityStartTime(a);
14                 c_0a_end = getActivityEndTime(a);
15                 {c'_0a_start, c'_0a_end} = f(c_0a_start, c_0a_end);
16                 if (c'_0a_start ⊆ c_0a_start) and (c'_0a_end ⊆ c_0a_end) then
17                     c_0a_start = c'_0a_start;
18                     c_0a_end = c'_0a_end;
19                 end
20             end
21         end
22         C = propagate((V, A, Γ));
23     end
24 end
```

**Algorithmus 4** : Aktualisierung dynamischer Zeitbedingungen

### 6.2.3 Selektion von Kontrollpunkten

Die bisher beschriebene Vorgehensweise für die Ausführungsphase verwendet jedes Ereignis der Prozessinstanz (d. h. Start und Ende von Aktivitäten) als *Kontrollpunkt* für die Aktualisierung des Zeitmodells bzw. der Zeitdaten. Dies ist unnötig, da die freien Schedules garantieren, dass es zu keinerlei Zeitfehlern kommt, solange die Aktivitäten in den von den freien Schedules vorgeben Intervallen starten und enden sowie die vorgegebene Dauer einhalten. Es genügt daher, das Zeitmodell lediglich dann zu aktualisieren, wenn eine der Aktivitäten eines der für ihre Ausführung vorgegebenen Intervalle (Start, Ende, Abstand oder Dauer) nicht einhält. In diesem Fall muss geprüft werden, ob das aktuelle Zeitmodell der Prozessinstanz automatisch korrigiert werden kann, oder ob eine Eskalation durchgeführt werden muss.

Chen und Yang haben mehrere Methoden entwickelt [Chen & Yang, 2005a], [Chen & Yang, 2007] wie die Anzahl der Kontrollpunkte trotz Nichteinhaltung der vorgegebenen Intervalle weiter gesenkt werden kann. Hierzu teilen sie die Definition der Inkonsistenz in 3 Stufen auf, namentlich „schwach konsistent“, „schwach inkonsistent“ und „stark inkonsistent“. Diese geben an, wie stark die Inkonsistenz ausgeprägt ist, d. h., wie viel zeitlich geändert werden müsste, um wieder einen konsistenten Zustand zu erreichen. Basierend darauf reduzieren sie die Anzahl der Kontrollpunkte, indem sie die beiden neuen Zustände *schwach konsistent* und *schwach inkonsistent* bei der Auswahl der Kontrollpunkte differenzierter betrachten. Sie bewerten hierzu, um wie viel die Dauer der nachfolgenden Aktivitäten eingeschränkt werden müsste, um wieder einen konsistenten Zustand zu erreichen.

Allerdings lässt eine zu starke Reduktion der Kontrollpunkte zum einen die berechneten Zeitdaten für die Prognose nutzlos werden, zum anderen wird die Überlagerung mehrere Instanztypen in den Zeitdaten weiter bestehen, auch nachdem feststeht, welcher der Instanztypen im konkreten Fall vorliegt. Vor allem Letzteres stellt ein Problem dar, da, je nach verwendetem Ansatz zur Behandlung der alternativen Wege (vgl. Abschnitt 4.1.6 und 4.2.5), die Zeitdaten nicht in allen Instanztypen in gleichem Maße gültig sind oder zu große Beschränkungen für die Ausführungszeiten der Aktivitäten auferlegen. Daher muss zumindest nach dem Ausführen eines XOR-Split- oder eines ENDLOOP-Knotens ein Kontrollpunkt eingeführt werden, um die nicht mehr zur Ausführung kommenden Pfade aus dem Zeitmodell zu entfernen (siehe Beispiel 6.4).

Für die Prognose von Zeitdaten wiederum kann die Prozessinstanz fast nicht genug Kontrollpunkte enthalten. Hier gilt bis zu einem gewissen Grad, dass die Prognose um so genauer wird, je mehr Kontrollpunkte die Prozessinstanz enthält. Teilweise empfiehlt es sich daher, bei lang laufenden Aktivitäten weitere Kontrollpunkte während der Laufzeit einzufügen. Bei diesen kann beispielsweise eine Fortschrittsangabe der Aktivitäten dazu verwendet werden, die Dauer der Aktivitäten abzuschätzen, um genauere Zeitdaten zu erhalten.

Ein Aspekt, der beim Für und Wider der vorgestellten Selektions-Strategien mit in Betracht gezogen werden muss, ist der nicht unerhebliche Rechenaufwand, welcher für das Aktualisieren des Zeitmodells benötigt wird. Es empfiehlt sich daher, mehrere Selektions-Strategien anzubieten, die entweder auf Prozessbasis oder durch die Konfiguration des Prozess-Management-Systems ausgewählt werden können. Damit lässt sich der für den konkreten Fall nötige Grad der Genauigkeit vs. Rechenaufwand einstellen. Als Kompromiss kann eine weitere Strategie angeboten werden, bei der ein zeitlicher Maximal- und/oder Minimal-Abstand zwischen den einzelnen Kontrollpunkten besteht. So kann der für eine ausreichend gute Prognose nötige Aktualisierungsgrad bei gleichzeitiger Reduktion des Rechenaufwandes erreicht werden.

## 6.3 Behandlung von Zeitfehlern

Wird eine Aktivität nicht im vorgegebenen Zeitintervall gestartet bzw. beendet, wird das Zeitmodell der Prozessinstanz inkonsistent, wodurch diese nicht mehr normal fortgesetzt werden kann. In diesem Fall tritt ein *Zeitfehler* auf, den es zu behandeln gilt. Hierzu wird ein *Eskalations-Mechanismus* gestartet, welcher zunächst selbstständig versucht den Zeitfehler zu beheben. Gelingt dies nicht, wird entweder der für die entsprechende Aktivität zuständige *Agent* oder eine verantwortliche Person benachrichtigt. Diese ist nun dafür zuständig den Zeitfehler entweder zu beheben oder die Prozessinstanz abzubrechen. Um den Agenten bei seiner Aufgabe zu unterstützen, können ihm verschiedene Vorschläge zur Behebung des Zeitfehlers unterbreitet werden. Einige dieser Möglichkeiten werden im Folgenden vorgestellt, wobei diese nach der Art des Zeitfehlers, d. h. nach dem Zeitzustand nach der Neubewertung, gegliedert sind.

**ACTIVATED - LATE** Die Aktivität hätte aufgrund eines vorgegebenen oder impliziten Maximal-Abstandes bereits gestartet werden müssen. Dies ist bisher jedoch ausgeblieben. In diesem Fall sollten zunächst die Bearbeiter der entsprechenden Aktivität (und möglicherweise auch deren Vorgesetzten) über dieses Problem benachrichtigt werden. Danach besteht beispielsweise die Möglichkeit, die maximale Dauer der verspäteten Aktivität zu begrenzen, um die verlorene Zeit wieder einzuholen. Andere Möglichkeiten sind die Maximal-Dauer einer der nachfolgenden Aktivitäten oder einen der Minimal-Abstände zwischen den nachfolgenden Aktivitäten zu verkürzen. Ist dies nicht möglich, muss entweder der die Aktivität einschließende Maximal-Abstand verlängert oder ein Termin nach hinten verschoben werden.

**STARTED/SUSPENDED - LATE** Die Ausführung der Aktivität dauert bereits länger als die spezifizierte Maximal-Dauer. In diesem Fall muss die Maximal-Dauer der Aktivität erhöht werden. Dies macht eventuell weitere Änderungen nötig, um ein konsistentes Zeitmodell zu erhalten (z. B. Verschieben von Terminen nach hinten, Reduktion der maximalen Dauer anderer Aktivitäten, etc.). Außerdem

sollte der ausführende Agent benachrichtigt werden, damit dieser die Aktivität schnellstmöglich beendet, um so die nötigen Änderungen zu minimieren.

**COMPLETED - EARLY** Die Aktivität wurde zu früh beendet. Das heißt, sie hat ihre spezifizierte Minimal-Dauer unterschritten, wodurch Maximal-Abstände zu beziehungsweise zwischen nachfolgenden Aktivitäten eventuell nicht mehr eingehalten werden können. Das Unterschreiten der Minimal-Dauer stellt meist kein Problem dar, jedoch sollte der Benutzer darüber benachrichtigt werden, da es beispielsweise darauf hindeuten kann, dass ein Teilschritt der Aktivität vergessen wurde. Können jedoch zusätzlich Maximal-Abstände nicht mehr eingehalten werden, müssen diese entweder vergrößert oder umgebende Minimal-Abstände verlängert bzw. Termine nach vorne verlegt werden.

Die Fehlerzustände NOT_ACTIVATED - LATE und COMPLETED - LATE lassen sich auf die Fehlerzustände ACTIVATED/STARTED/SUSPENDED - LATE einer vorhergehenden Aktivität zurückführen und werden bei deren Korrektur ebenfalls angepasst. Die Zustandskombination STARTED/SUSPENDED - EARLY bedeutet lediglich, dass es – falls die Aktivität beendet würde – zum Fehlerzustand COMPLETED - EARLY kommt. Ähnlich verhält es sich mit dem Zustand ACTIVATED - EARLY. Hier sollte die Aktivität, wie in Abschnitt 6.2 beschrieben, für die Ausführung gesperrt bzw. entsprechend in der Arbeitsliste markiert werden. Ansonsten wird, sollte die Aktivität jetzt gestartet werden, zum einen ein Minimal-Abstand unterschritten (worauf der Benutzer von der Ausführungskomponente hingewiesen werden sollte) und zum anderen kommt es beim Beenden der Aktivität eventuell zum Fehlerzustand COMPLETED - EARLY. Eine Übersicht über die Zeitfehler und ihre Behandlungsmöglichkeiten ist in Tabelle 6.2 zu finden.

Die vorgestellten Möglichkeiten zur Behebung von Zeitfehlern sind nicht vollständig. So ist es beispielsweise immer dann alternativ möglich eine Aktivität aus der Prozessinstanz zu entfernen, wenn die maximale Dauer einer Aktivität oder ein minimaler Abstand verkürzt werden muss. Häufig ist es auch möglich, Zeitfehler zu beheben, indem die Reihenfolge der Aktivitäten verändert wird. Hierzu werden beispielsweise Aktivitäten vorgezogen, deren Eingangsbedingungen für den Normalfall durch den Kontrollfluss noch nicht erfüllt sind [van der Aalst *et al.*, 2005].

Ein weiterer wichtiger Punkt bei der Behandlung von Zeitfehlern ist, dass durch die oben vorgeschlagenen Möglichkeiten zur Behebung der Fehler häufig die Termine und Zeitbedingungen anderer Aktivitäten beeinflusst werden. Daher reicht es zur Lösung des Problems meist nicht, nur eine Aktivität zu verändern, sondern es muss oft eine ganze Reihe von Aktivitäten und Terminen verändert werden. Auch kann das System meist nicht entscheiden, welche der Alternativen zu wählen ist, da hierzu das nötige Hintergrundwissen fehlt. Beispielsweise kann nicht jede Aktivität einfach verkürzt oder verlängert werden. Daher wird beim Auftreten eines Zeitfehlers meist ein Benutzer eingreifen müssen, um das

| Fehlerzustand | Behandlungsmöglichkeiten |
|---|---|
| NOT_ACTIVATED - LATE | • Eine andere Aktivität besitzt einen der Zustände STARTED/SUSPENDED/ACTIVATED - LATE. Nach der Korrektur des Zeitfehlers der anderen Aktivität gilt auch für diese Aktivität wieder ein erlaubter Zeitzustand. |
| ACTIVATED - EARLY | • Die Aktivität wird durch die Ausführungskomponente gesperrt, bis zum entsprechenden Wechsel auf ACTIVATED - ON_TIME. |
| ACTIVATED - LATE | • Verkürzen der Maximal-Dauer einer Aktivität<br>• Verkürzen eines Minimal-Abstandes<br>• Verlängern der umgebenden Maximal-Abstände<br>• Verschieben eines Termins nach hinten |
| STARTED/SUSPENDED - LATE | • Verlängern der maximalen Dauer der Aktivität |
| COMPLETED - EARLY | • Verlängerung eines nachfolgenden Maximal-Abstandes<br>• Verkürzen eines umgebenden Minimal-Abstandes<br>• Verschieben eines Termines nach vorne |
| COMPLETED - LATE | • Wird bereits durch die Korrektur des vorhergehenden Fehlerzustandes STARTED/SUSPENDED - LATE behoben. |

Tabelle 6.2: Zeitfehler und Behandlungsmöglichkeiten

System bei der Behebung des entsprechenden Problems zu unterstützen. Da dies jedoch sehr komplex sein kann, sollte dem Benutzer an dieser Stelle möglichst viel Unterstützung durch das System zukommen. Dies kann beispielsweise in der Form geschehen, dass das System dem Benutzer mögliche Varianten zur Behebung des Problems vorschlägt und dieser eine von diesen auswählt. Sollte keine der vorgeschlagenen Varianten im konkreten Fall anwendbar sein, bleibt dem Benutzer entweder die Möglichkeit, die Prozessinstanz durch dynamische Modifikationen so zu verändern, dass der Fehler dadurch behoben wird, oder er muss die gesamte Prozessinstanz abbrechen, da eine korrekte Ausführung nicht mehr möglich ist.

Eine Möglichkeit Zeitfehler zu vermeiden ist die Einführung weiterer Zeitzustände. Damit kann der Benutzer durch einen geeigneten Notifikationsmechanismus (beispielsweise seine Arbeitsliste) vor Ablauf der Zeit auf einen drohenden Zeitfehler hingewiesen und somit in die Lage versetzt werden, korrigierend gegenzusteuern, um diesen zu verhindern oder zumindest die Auswirkungen zu minimieren. Ein Beispiel für einen solchen weiteren Zeitzustand wäre RECOMMENDED innerhalb des Zeitzustandes ON_TIME. Für diesen könnte ein gewisser Prozentsatz des Intervalls von ON_TIME gewählt oder eine weitere

Zeit definiert werden. Sobald dieser Zeitzustand verlassen wird, kann der Benutzer durch geeignete Mechanismen hierauf aufmerksam gemacht werden. Er ist damit z. B. in der Lage, die entsprechende Aktivität mit erhöhter Priorität zu behandeln.

Um die Ausführungsphase und den beschriebenen Mechanismus zur Fehlerbehandlung ausreichend unterstützen zu können, ist ein entsprechend ausgelegtes Laufzeitsystem nötig. Dieses wird im nächsten Abschnitt beschrieben.

## 6.4 Das Laufzeitsystem

In Abbildung 6.1 ist eine schematische Übersicht der Kern-Komponenten eines Prozess-Management-Systems, mit dem Zeitmanagement als Schwerpunkt, zu sehen.

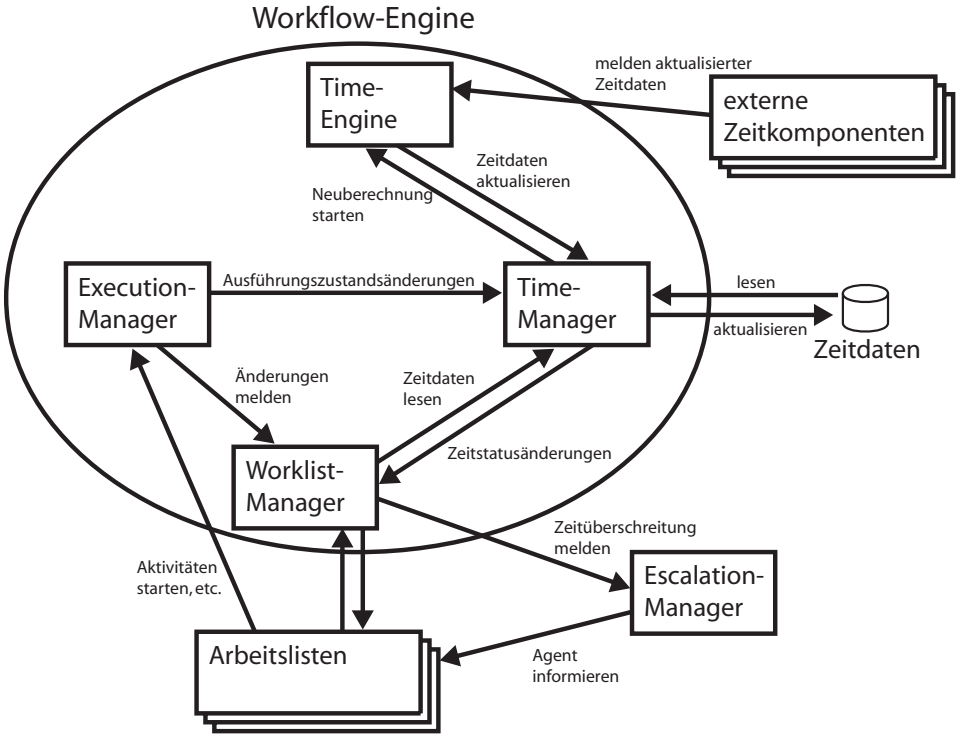



Abbildung 6.1: Laufzeitsystem - schematisch Übersicht

Die für die Ausführung der Prozessinstanzen zuständige Workflow-Engine gliedert sich in die folgenden, für das Zeitmanagement wichtigen Komponenten:

**Execution-Manager** Der *Execution-Manager* ist die zentrale Komponente der Prozessausführung. Er ist für die gesamte Verwaltung und Steuerung der Prozessinstanzen sowie für die Ausführung ihrer Aktivitäten verantwortlich.

**Time-Manager** Der *Time-Manager* kümmert sich um die Verwaltung der zu den Prozessinstanzen gehörenden Zeitdaten. Diese werden unter anderem bei Änderungen des Ausführungszustands (z. B. von ACTIVATED nach STARTED) aktualisiert. Darauf basierend überwacht der Time-Manager die Zeitzustände der Aktivitäten und meldet deren Änderungen (z. B. von ON_TIME nach LATE) an den Worklist-Manager.

**Time-Engine** Die *Time-Engine* berechnet die aktualisierten Zeitdaten der Prozessinstanzen. Sie wird dazu vom Zeit-Manager angestoßen und führt daraufhin die Zeitalgorithmen aus. Dabei greift sie auf die von den externen Zeitkomponenten bereitgestellten Funktionen zu, um die dynamischen Zeitbedingungen der Instanz zu ermitteln. Die Time-Engine kann zusätzlich von einer externen Zeitkomponente angestoßen werden, falls sich bei dieser eine Veränderung der Zeitdaten ergeben hat (z. B. durch das Schreiben des entsprechenden Datenelementes).

**Worklist-Manager** Der *Worklist-Manager* ist für die Bereitstellung und Aktualisierung der Arbeitslisten verantwortlich. Er greift außerdem auf den Zeit-Manager zu, um die aktuellen Zeitdaten der angebotenen und zukünftigen Aktivitäten zu erhalten. Letzteres kann er beispielsweise dazu verwenden, dem Benutzer eine Prognose für die nahe Zukunft zu bieten (siehe Abschnitt 6.6).

Um die Workflow-Engine herum gruppieren sich weitere Komponenten, welche vor allem zur Kommunikation mit der Workflow-Engine dienen:

**Escalation-Manager** Der *Escalation-Manager* dient dem Prozess-Management-System dazu, drohende oder tatsächlich aufgetretene Zeitfehler zu behandeln. Dazu wird er vom Worklist-Manager über Zeitfehler informiert und entscheidet daraufhin über eine adäquate Reaktion (siehe Abschnitt 6.3).

**Arbeitslisten** Jeder Agent besitzt eine eigene *Arbeitsliste*, welche ihm die Interaktion mit dem Prozess-Management-System ermöglicht. Die Arbeitsliste bietet dem Agenten seine Aktivitäten zur Bearbeitung an. Außerdem stellt sie dem Agenten weitere Informationen über die anstehenden Aktivitäten wie minimale und maximale Dauer zur Verfügung, um diesen bei der Wahl der nächsten Aktivität(en) zu unterstützen. Entscheidet sich ein Agent für eine der angebotenen Aktivitäten, ermöglicht die Arbeitsliste deren Start und überwacht ihre lokale Ausführung. Des Weiteren stellt die Arbeitsliste eine Möglichkeit für das Prozess-Management-System dar, den Agenten über drohende Zeitfehler zu informieren.

**Externe Zeitkomponenten** *Externe Zeitkomponenten* stellen die Schnittstelle dar, über die beispielsweise *Terminplanungssysteme* an die Zeitplanung des Prozess-Management-Systems angebunden werden können. Allgemein stellt eine externe Zeitkomponente eine Funktion bereit, über welche die Werte für eine dynamische Zeitbedingung ermittelt werden können (z. B. durch die Time-Engine).

Die folgenden beiden Abschnitte diskutieren, welche Anforderungen die dynamische Modifizierbarkeit von Prozessinstanzen zur Ausführungszeit und die Prognose von Zeitdaten für zukünftige Aktivitäten an diese Architektur stellen und welche Lösungsmöglichkeiten sich hierfür bieten.

## 6.5 Dynamische Modifikationen

Für eine flexible Gestaltung von Prozessen ist es wichtig, dass diese zur Laufzeit modifizierbar sind, um beispielsweise auf spezielle Situationen reagieren zu können, die während der Modellierung des Prozessschemas nicht eingeplant wurden. Diese *dynamischen Änderungen* lassen sich nach [Reichert & Dadam, 1997], [Reichert & Dadam, 1998] grundsätzlich zwei Gruppen von Operationen zuordnen, den *status-* und den *strukturändernden* Operationen.

**Statusändernde Operationen** finden beispielsweise bei der Änderung von Ausführungsparametern (z. B. Hinzufügen/Löschen eines Parameters) oder Zuständen (z. B. Zurücksetzen) statt. Sie modifizieren den aktuellen Zustand von Kanten und Knoten oder den Inhalt von Datenelementen.

**Strukturändernde Operationen** hingegen verändern, wie ihr Name schon sagt, die Struktur des Prozessgraphen. Das heißt, sie fügen der *Prozessinstanz* z. B. weitere Aktivitäten und Kanten hinzu.

Die beiden Modifikationsarten können meist nicht getrennt betrachtet werden. Zum Beispiel müssen bei Änderungen des Prozessgraphen auch die aktuellen Ausführungszustände der Aktivitäten mit beachtet werden, um fehlerhafte oder unsinnige Operationen zu vermeiden. Das Einfügen einer Aktivität in einen bereits ausgeführten Bereich ist beispielsweise nicht möglich [Reichert, 2000] [Rinderle *et al.*, 2004c].

Auch im Rahmen des Zeitmanagements müssen beide Modifikationsarten betrachtet werden. Einerseits beeinflussen Änderungen an Zeitbedingungen, beispielsweise das Ändern eines Termins, den Status von Aktivitäten, während andererseits das Einfügen oder Löschen einer weiteren Zeitkante die Struktur der Prozessinstanz verändert.

### 6.5.1 Statusändernde Operationen

Statusändernde Operationen finden während der Ausführung einer Prozessinstanz häufig statt. Beispielsweise ist für jedes Starten oder Beenden einer Aktivität eine Statusänderung

nötig, um den Ausführungszustand der Aktivität zu aktualisieren. Auch das Schreiben eines Datenelements durch eine Aktivität stellt eine solche statusändernde Operation dar. Welche Aktionen vom Zeitmanagement bei einer Änderung des Ausführungszustandes durchzuführen sind, wurde bereits in Abschnitt 6.2 beschrieben. Die Änderung eines Datenelements hat auf das Zeitmanagement zunächst keine direkten Auswirkungen. Es kann jedoch dadurch zu indirekten Auswirkungen kommen, dass eine der dynamischen Zeitbedingungen von einem veränderten Datenelement abhängig ist. Auch die hierbei nötigen Schritte wurden bereits in Abschnitt 6.2.2 diskutiert.

Es bleibt daher vor allem noch eine statusändernde Operation zu betrachten, welche für das Zeitmanagement wichtig ist: das Zurücksetzen von Aktivitäten oder ganzen Teilen der Prozessinstanz. Die Rücknahme noch laufender oder bereits beendeter Aktivitäten kann nach [Reichert, 2000] aus verschiedenen Gründen notwendig sein.

- Aufgrund des Fehlschlags einer Aktivität soll der Ablauf in einen früheren Zustand zurückgesetzt werden.

- Ein Benutzer möchte bereits gestartete oder beendete Aktivitäten zurücknehmen beziehungsweise erneut ausführen.

Ein Zurücksetzen schließt die Rücknahme sämtlicher Änderungen der Prozessinstanz, die Wiederherstellung des alten Datenbestandes und möglicherweise die Kompensation der Auswirkungen einzelner Aktivitäten (sofern möglich) mit ein.

Aus der Perspektive des Zeitmanagements werden für das Zurücksetzen von Aktivitäten zunächst einfach deren Ausführungszeitpunkte gelöscht. Bei Aktivitäten oder Kanten mit dynamischen Zeitbedingungen müssen danach die zum damaligen Zeitpunkt gültigen Werte der Zeitbedingungen wiederhergestellt werden. Hierfür ist eine Ausführungshistorie vonnöten, in der jegliche Änderungen an dynamischen Zeitbedingungen für eine spätere Rücknahme protokolliert werden. Zusätzlich muss es möglich sein, die externen Zeitkomponenten auf ihren früheren Zustand zurückzusetzen.

Nachdem auf diese Weise ein früherer Zustand wiederhergestellt wurde, müssen die neuen Zeitbedingungen durch erneute Ausführung der Algorithmen bestimmt werden. Da sich die Zeit selbst nicht zurücknehmen lässt, kann es hierbei vorkommen, dass die Prozessinstanz als inkonsistent eingestuft wird. In diesem Fall müssen die normalen Schritte zur Behandlung eines Zeitfehlers (siehe Abschnitt 6.3) durchgeführt werden.

### 6.5.2 Strukturändernde Operationen

Strukturändernde Operationen sind ein besonderes Feature von ADEPT [Reichert & Dadam, 1998] [Reichert *et al.*, 2005] [Weber *et al.*, 2007]. Dabei sind verschiedene Änderungen möglich, deren Auswirkungen auf das Zeitmanagement im Folgenden beschrieben werden.

**Einfügen von Aktivitäten** Nach dem Einfügen einer neuen Aktivität in die Prozessinstanz, sofern aufgrund der Kriterien von ADEPT möglich, muss diese entsprechend der Transformation ebenfalls in das Zeitmodell eingefügt werden. Danach können die Standard-Algorithmen zur Berechnung der neuen Zeitdaten herangezogen werden.

**Löschen von Aktivitäten** Das automatische Löschen eines Knotens kann aus verschiedenen Gründen zu Problemen führen. In Verbindung mit Zeit wird es dann kompliziert, wenn die Aktivität der Start oder das Ziel einer Zeitkante ist. In diesem Fall ist nicht eindeutig, was mit dieser Zeitkante zu geschehen hat. Möglichkeiten wären das Löschen der Zeitkante oder das Umsetzen auf eine andere Aktivität. An dieser Stelle ist es nötig, den Benutzer mit in den Änderungsprozess einzubeziehen. Dieser muss dann die Prozessinstanz zunächst so verändern, dass die zu löschende Aktivität keine aus- oder eingehenden Zeitkanten mehr besitzt. Dabei ist eine Unterstützung durch das System hilfreich, indem es dem Benutzer die Konsequenzen seines Handelns aufzeigt.

Besitzt ein Knoten keine ein- oder ausgehenden Zeitkanten (mehr), kann er einfach aus der Prozessinstanz und dem Zeitmodell gelöscht werden. Dabei werden dann im Zeitmodell auch alle Constraints gelöscht, deren Start oder Ziel die Knoten der Aktivität waren. Danach können wieder die Standard-Algorithmen zur Berechnung der neuen Zeitdaten herangezogen werden.

**Einfügen von Zeitkanten** Das Einfügen einer Zeitkante stellt lediglich eine weitere Einschränkung der Zeitbedingungen dar. Diese ist, wie in Abschnitt 6.2.1 diskutiert, durch die Algorithmen einfach umsetzbar.

**Löschen von Zeitkanten** Das Löschen einer Zeitkante stellt ebenfalls kein Problem für die Algorithmen dar. Jedoch sind in diesem Fall die in Abschnitt 6.2.1 diskutierten Optimierungen nicht mehr anwendbar, da es zu einer Erweiterung der Zeitbedingungen kommt.

Diese Operation tritt relativ häufig auf, da sie beispielsweise dann vonnöten ist, wenn der Benutzer absichtlich eine Zeitbedingung (z. B. einen Mindestabstand) ignoriert.

**Modifikation der Ausführungsreihenfolge** Das Überspringen von Aktivitäten kann an verschiedenen Stellen interessant sein. Beispielsweise kann es dazu genutzt werden, eine Schlüsselaktivität vorzuziehen, deren Termin in Gefahr ist, um diesen doch noch zu halten. Das heißt, eine Aktivität wird ausgeführt, obwohl nicht alle ihre Eingangsbedingungen erfüllt sind. Das ADEPT-Basismodell bietet dazu verschiedene Arten des Überspringens von Aktivitäten an [Reichert, 2000] [Reichert *et al.*, 2003]

- Aktivität überspringen ohne Nachholen.

- Aktivität überspringen mit Nachholen der übersprungenen Schritte.
- Vorziehen von Schritten.

Hierbei bestehen, sofern die Modifikation der Ausführungsreihenfolge aufgrund der Kriterien von ADEPT überhaupt möglich ist, in Verbindung mit Zeit ähnliche Probleme wie beim Löschen von Aktivitäten. Sofern eine der verschobenen bzw. übersprungenen Aktivitäten Start oder Ziel einer Zeitkante ist, kann es durch diese Operation beispielsweise dazu kommen, dass die Eindeutigkeit der Ablauffolge (siehe Abschnitt 4.1.3.2) nicht mehr gewährleistet ist, da der Zielknoten einer Zeitkante Vorgänger ihres Quellknotens ist. In diesem Fall muss die Operation zunächst unterbrochen werden, da ein Eingreifen des Benutzers nötig ist. Dieser muss entscheiden, ob die entsprechende Zeitkante (d. h. die durch sie ausgedrückte Zeitbedingung) gelöscht werden kann oder ob die Operation rückgängig gemacht werden muss.

Nachdem die Modifikation der Ausführungsreihenfolge erfolgreich durchgeführt wurde, muss das minimale Netz der Prozessinstanz neu erstellt werden. Eine Modifikation des minimalen Netzes ist nur schwer möglich, da zur Bestimmung der Auswirkungen auf die ermittelten impliziten Constraints eine aufwendige Graphanalyse nötig ist (vgl. Abschnitt 6.2.1).

Es ist möglich, mehrere Änderungen nacheinander durchzuführen, zum Beispiel das Löschen von Zeitkanten gefolgt vom Löschen einer Aktivität, und erst zum Schluss das Zeitmodell und die Zeitdaten zu aktualisieren. Dabei müssen dann die Besonderheiten bei der Aktualisierung aller durchgeführten Änderungen beachtet werden.

Außerdem kann es bei sämtlichen der vorgestellten Änderungen während der Aktualisierung der Zeitdaten dazu kommen, dass Inkonsistenzen bzw. Zeitfehler aufgedeckt werden. Diese können jedoch auf normale Weise durch die bereits vorgestellten Möglichkeiten behandelt werden (siehe Abschnitte 4.2.4.1 und 6.3).

## 6.6 Prognose von Zeitdaten

Ein interessantes Anwendungsgebiet des Zeitmanagements, neben der Überwachung von Zeitbedingungen, ist die Prognose von Start- und Endzeitpunkten zukünftiger Aktivitäten. Da Bearbeiter immer nur einzelne Schritte sehen, fehlt ihnen der Überblick über zukünftige Prozessschritte aber auch über alle laufenden Prozesse, in denen sie involviert sind. Häufig sind die Benutzer eines PMS daher „überrascht“, welche Aktivitäten sie als Nächstes bearbeiten sollten. Überrascht in dem Sinne, dass die Aktivitäten plötzlich ohne Vorankündigung in ihrer Arbeitsliste auftauchen, sobald ihre Eingangsbedingungen erfüllt sind. Jedoch stehen Informationen über zukünftige Aktivitäten dem Prozess-Management-System eigentlich viel früher zur Verfügung. Wird beispielsweise die erste Aktivität einer

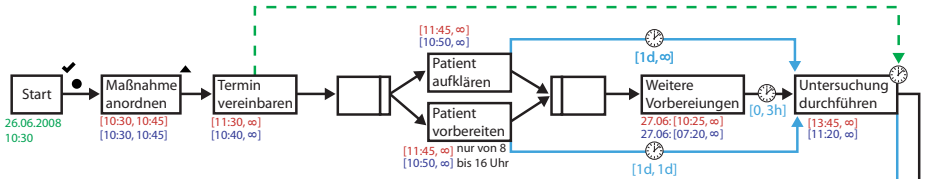
Sequenz gestartet, so ist klar, dass die nachfolgenden Aktivitäten auch „bald“ für die Ausführung bereitstehen.

Die Aussage „bald“ ist jedoch eine sehr ungenaue Zeitangabe. Es kann sich dabei um wenige Minuten, aber im Fall lang laufender Prozesse auch um mehrere Wochen oder Monate handeln. Es ist allerdings wenig sinnvoll, den Benutzer bereits auf Aktivitäten hinzuweisen, welche er erst in einigen Wochen bearbeiten kann bzw. soll. An diesem Punkt sind das Zeitmanagement beziehungsweise die Zeitdaten der *Prozessinstanzen* von nutzen. Mithilfe dieser ist es möglich, die Aussage „bald“ zu konkretisieren und dem Benutzer so einen Anhaltspunkt zu geben, welche Aufgaben ihn in den nächsten Stunden oder Tagen erwarten (ähnlich einem Kalendersystem). Hierzu werden die Startzeitintervalle der nächsten Aktivitäten der Prozessinstanz untersucht, um so herauszufinden, ob die Aktivität in einem vom Benutzer spezifizierten Zeitrahmen (z. B. innerhalb der nächsten 8 Stunden) zur Ausführung bereitstehen wird.

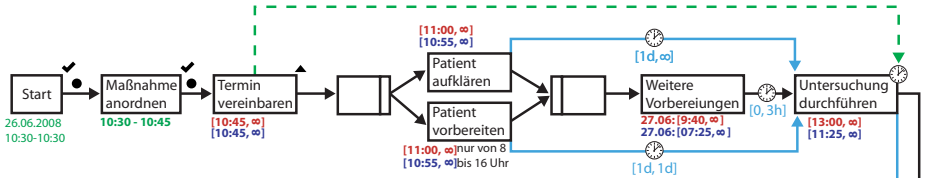
Dabei gilt es einige Aspekte zu beachten: Es müssen die in Abschnitt 4.2.4.2 (Seite 92) beschriebenen Eigenschaften der errechneten Menge der *freien Schedules* berücksichtigt werden, denn diese geben lediglich Intervalle an, in welchen die Ausführbarkeit der gesamten Prozessinstanz garantiert werden kann. Benötigt eine Aktivität weniger Zeit als die angegebene Maximal-Dauer, wird sich der von den freien Schedules angegebene früheste Startzeitpunkt nachfolgender Aktivitäten nach vorne verschieben. An dieser Stelle sollten, damit der Benutzer nicht wieder „überrascht“ wird, auch die Werte des minimalen Netzes hinzugezogen werden, welche alle Eventualitäten (im Rahmen der erlaubten Zeitbedingungen) mit einbeziehen. Das heißt, es werden beispielsweise auch Zeitpunkte für die Aktivitäten beachtet, die nur dann Vorkommen, wenn alle vorhergehenden Aktivitäten nicht länger als ihre Minimal-Dauer benötigt haben. Beispiel 6.5 verdeutlicht dies.

Außerdem ist an dieser Stelle ein in Abschnitt 6.2.3 angesprochener Aspekt von Bedeutung. Weist eine Aktivität eine sehr große Differenz zwischen maximaler und minimaler Dauer auf, werden auch die nachfolgenden Aktivitäten eine große Differenz zwischen frühestem und spätestem Startzeitpunkt aufweisen. Damit besteht wieder das Problem, dass der Benutzer auf Aufgaben aufmerksam gemacht wird, die er erst viel später bearbeiten kann. Um dies zu relativieren, bietet es sich an während der Ausführung der langlaufenden bzw. der Aktivität mit großem Spielraum weitere Kontrollpunkte einzuführen, mithilfe derer die reale Dauer der Aktivität besser abgeschätzt werden kann. Damit können dem Benutzer dann detailliertere Informationen zur Verfügung gestellt werden.

Neben der eben diskutierten Vorausschau für den Benutzer lassen sich die vom Zeitmanagement vorhergesagten Zeitdaten noch für einige andere Dinge verwenden. [Eder *et al.*, 2003] erörtert beispielsweise die Möglichkeit, auf Basis der vorhergesagten Zeitdaten für jeden Benutzer einen *persönlichen Terminplan* zu berechnen. Dieser empfiehlt dem Benutzer, wann und in welcher Reihenfolge er seine Aufgaben bearbeiten sollte, um einerseits die zur Verfügung stehende Zeit optimal auszunutzen und andererseits Verzögerungen beziehungsweise Zeitüberschreitungen zu vermeiden. Außerdem werden




Das erste Bild zeigt eine Prozessinstanz vor Ausführung der Aktivität Maßnahme anordnen. Dabei sind die Werte der freien Schedules in Rot, die Werte des Zeitmodells in Blau sowie durch die Ausführung festgelegte Werte in Grün eingetragen. Wie ersichtlich, hat die Aktivität Termin vereinbaren vor der Ausführung der Aktivität Maßnahme anordnen bei den freien Schedules den Wertebereich $[26.06\ 11{:}30, \infty]$ und im Zeitmodell den Wertebereich $[26.06\ 10{:}40, \infty]$.




Nach Ausführung der Aktivität Maßnahme anordnen ändern sich sowohl der Wertebereich der freien Schedules als auch der des Zeitmodells für die Aktivität Termin vereinbaren. Jedoch verschiebt sich beim Zeitmodell der früheste Zeitpunkt lediglich nach hinten ($[26.06\ 10{:}40, \infty] \rightarrow [26.06\ 10{:}45, \infty]$), wohingegen sich der früheste Zeitpunkt der freien Schedules nach vorne verschiebt ($[26.06\ 11{:}30, \infty] \rightarrow [26.06\ 10{:}45, \infty]$). Dadurch könnte die Aktivität also früher ausgeführt werden als ursprünglich vorhergesagt. Wird lediglich die Menge der freien Schedules für die Vorhersage verwendet, könnte der Benutzer durch das frühere Erscheinen der Aktivität Termin vereinbaren überrascht werden.

Beispiel 6.5: Zeitmodell und freie Schedules

verschiedene Scheduling-Strategien wie Earliest-Deadline-First[1], Most-Probable-Deadline-Violation[2] und Lowest-Proportional-Slack[3] [Eder *et al.*, 2006] auf ihre Eignung untersucht, in Kombination mit einem persönlichen Terminplan, Zeitüberschreitungen zu vermeiden.

Eine weitere in [Zhao & Stohr, 1999] untersuchte Möglichkeit zur Vermeidung von Zeitüberschreitungen ist die dynamische *Priorisierung* der Aufgaben anhand der noch verbleibenden Zeit. Dabei wird zusätzlich die aktuelle und für die Zukunft vorhergesagte Lastsituation für die Bearbeitung einzelner Aktivitäten beachtet. Auf Basis dieser Werte

[1] engl.: frühester Termin zuerst
[2] engl.: wahrscheinlichste Terminverletzung zuerst
[3] engl.: kleinster verhältnismäßiger Schlupf zuerst

werden dynamisch die Priorität einzelner Aufgaben erhöht und diese der aktuellen Priorität entsprechend in der Arbeitsliste hervorgehoben. Damit sollen die Benutzer dazu bewegt werden, höher-priorisierte Aufgaben bevorzugt zu bearbeiten, um auf diese Weise mögliche Zeitüberschreitungen zu vermeiden.

[Panagos & Rabinovich, 1997] beschäftigt sich damit, wann, aufgrund der aktuellen Termine und der aktuellen Lastsituation, eine Zeitüberschreitung einer nachfolgenden Aktivität wahrscheinlich erscheint. Erhält man dabei für eine der Aktivitäten eine erhöhte Wahrscheinlichkeit, werden die Kosten, die eine sofortige Eskalation verursachen würde, mit den Kosten einer Eskalation zu einem späteren Zeitpunkt verglichen. Ergeben sich hierbei beträchtliche Einsparungen, so ist der Vorschlag in [Panagos & Rabinovich, 1997], lieber frühzeitig und nicht erst wenn nicht mehr vermeidbar, Eskalation durchzuführen. Diese ist möglicherweise unnötig, dafür aber kostengünstiger.

Weitestgehend unabhängig von Hilfestellungen für den Benutzer können die vorhergesagten Zeitdaten auch für ein zusätzliches *Ressourcenmanagement* verwendet werden. Dazu werden die Aktivitäten um die Information angereichert, welche Ressource, z. B. Maschinen, Räume, Werkzeuge oder Personal für die Ausführung benötigt werden. Auf Basis dieser Daten und der vorhergesagten Zeitdaten kann dann das Ressourcenmanagement für die Dauer der Ausführung der Aktivität die entsprechenden Ressourcen reservieren. Dadurch kann eine konfligierende Verwendung von Ressourcen durch verschiedene Aktivitäten zum selben Zeitpunkt vermieden und zugleich eine optimale Auslastung der zur Verfügung stehenden Ressourcen erreicht werden. Hierfür ist vor allem die berechnete Menge der freien Schedules hilfreich, da diese im angegebenen Intervall unabhängig von der Dauer der übrigen Aktivitäten die Ausführbarkeit der Aktivität garantieren. Somit kann das Ressourcenmanagement die Ressource ab einem beliebigen Zeitpunkt dieses Intervalls reservieren, ohne dabei auf andere Aktivitäten achten zu müssen. Nachdem das Ressourcenmanagement die Ressourcen ab einem Zeitpunkt reserviert hat, sollte das Zeitmodell derart aktualisiert werden, dass das Startzeitintervall auf diesen Zeitpunkt begrenzt wird. Dadurch können frühzeitig mögliche Auswirkungen dieser Reservierung auf die Startzeitpunkte der übrigen Aktivitäten erkannt werden. Zugleich ist es möglich, Ressourcenengpässe zu erkennen und durch geeignete Eskalationsmechanismen zu behandeln.

## 6.7 Zusammenfassung

Im letzten Kapitel sind wir darauf eingegangen, welche Vorgänge während der Ausführung einer Prozessinstanz stattfinden müssen und welchen Nutzen man durch die Verwendung des Zeitmanagements erzielen kann.

Dazu wurde neben den speziellen, während der Instanzierungs- und Ausführungsphase nötigen Schritten auch diskutiert, welche Optimierungen für die Algorithmen während der Ausführung einer Prozessinstanz möglich sind und wo deren Grenzen liegen. Außerdem

wurde betrachtet, wie die Zeitrechnung erweitert werden muss, um die Aktualisierung dynamischer Zeitbedingungen während der Ausführung zu unterstützen.

Da ein Benutzer nicht zur Einhaltung der Zeitbedingungen gezwungen werden kann, haben wir diskutiert, welche Möglichkeiten im Fall des Nichteinhaltens zur Rettung der Prozessinstanz bestehen, welche Unterstützung hier das Prozess-Management-System bieten kann und wo dessen Grenzen liegen und somit ein Mensch konsultiert werden muss.

Eng mit dieser Frage hängt die Art und Weise zusammen, wie das Zeitmanagement in das Prozess-Management-System integriert wird und welche Interaktionen hierbei stattfinden.

Ebenfalls wurde diskutiert, welche Einschränkungen für die Modifikation einer Prozessinstanz zur Laufzeit gelten und wie hiermit umgegangen werden kann.

Abschließend wurde erläutert, welche Prognosen für die Ausführungszeitpunkte der Aktivitäten aus den berechneten Werten abgleitet werden können und wie man diese Nutzen kann, um die Ausführung der Prozesse noch effizienter zu gestalten.

# 7 Ausgewählte Implementierungsaspekte

Im praktischen Teil der Diplomarbeit wurden die vorgeschlagenen Konzepte prototypisch implementiert. Dabei wurde das ADEPT2-System[1] als Basis verwendet [Reichert *et al.*, 2005], welches am Institut für Datenbanken und Informationssysteme der Universität Ulm entwickelt wird.

Die prototypische Implementierung unterstützt die Anreicherung von *Prozessschemata* um sämtliche der in Abschnitt 4.1 vorgestellte Zeitelemente. Darüber hinaus können auf Basis der modellierten Zeiteigenschaften eines Prozessschemas Zeitmodelle, inklusive deren minimaler Netze sowie Schedules für die *Prozessinstanzen* berechnet werden. Die von ADEPT2 bereitgestellte Programmierschnittstelle (API) wurde um die nötige zeitbezogene Funktionalität erweitert.

Im Folgenden wird zunächst der verwendete ADEPT2-Prototyp kurz vorgestellt (Abschnitt 7.1). Anschließend werden in Abschnitt 7.2 die Erweiterung der ADEPT2-API um zeitbezogene Funktionalität präsentiert und einige Details der Implementierung diskutiert. Abschnitt 7.3 beschäftigt sich mit der bereitgestellten Benutzerschnittstelle, über welche man sich über die aktuellen Zeitzustände und Ausführungszeitpunkte informieren kann. Abschließend folgt in Abschnitt 7.4 ein kurzes Fazit.

## 7.1 Der ADEPT2-Prototyp

Der Prototyp des ADEPT2-Prozess-Management-Systems basiert auf einer in der Programmiersprache Java[2] implementierten Client/Server-Architektur. Mithilfe der Clients findet die Modellierung der Prozesse sowie die Ausführung der zu den Prozessinstanzen gehörenden (manuellen) Aktivitäten statt. Der Server verwaltet die vorhandenen Prozessschemata und steuert die Ausführung von deren Instanzen (siehe Abbildung 7.1).

Für die Modellierung der Prozesse steht ein graphischer Editor zur Verfügung, der den Modellierer bei seiner Aufgabe unterstützt. Dieser erlaubt die Modellierung sämtlicher in Abschnitt 2.1.2 vorgestellte Konstrukte. Außerdem ermöglicht er die Anreicherung der Prozessschemata um weitere, für die Ausführung nötige, Informationen, wie beispielsweise die Spezifikation der Bearbeiterzuordnung einzelner Aktivitäten auf Grundlage eines Organisationsmodells. Ist ein Prozess vollständig spezifiziert, wird er dem Server übermittelt, welcher diesen fortan zur Ausführung anbietet.

[1] http://www.aristaflow.de Stand: 17.08.2008
[2] http://java.sun.com Stand: 17.08.2008

Der Benutzer interagiert mit einem ADEPT2-Server über seinen Client, welcher ihm das Starten neuer Prozessinstanzen sowie die Ausführung der ihm angebotenen Aktivitäten ermöglicht (siehe Abbildung 7.1). Möchte ein Benutzer eine neue Prozessinstanz starten, teilt der Client dies dem Server mit, welcher daraufhin das zugehörige Prozessschema lädt und zur Ausführung vorbereitet. Daraufhin bietet der Server den angemeldeten Clients, anhand der in den Aktivitäten des Prozessschemas hinterlegten Bearbeiterzuordnungsregeln, die zur Ausführung bereitstehenden Aktivitäten in Arbeitslisten an. Entscheidet sich der Benutzer für die Ausführung einer ihm angebotenen Aktivität, werden die hierfür benötigten Daten vom Server an den Client übertragen und die der Aktivität zugeordnete Anwendung beim Client gestartet. Nach dem Beenden der Aktivität überträgt der Client die von dieser gesammelten Daten zurück an den Server, welcher sich um deren Speicherung und Weiterverteilung an die übrigen Aktivitäten kümmert.

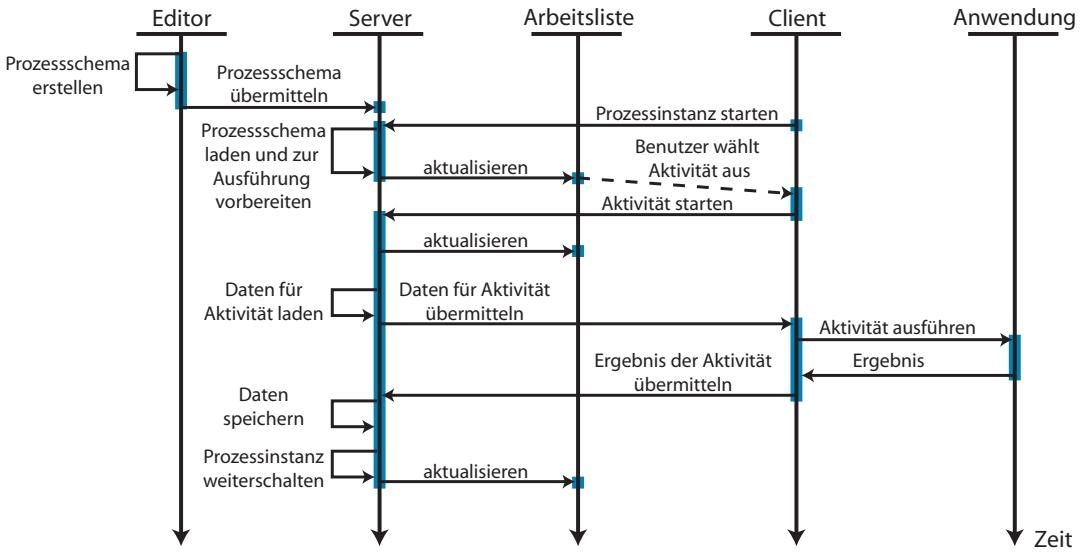



Abbildung 7.1: Interaktionsdiagramm einer Prozessausführung in ADEPT2

ADEPT2 unterstützt an Zeitinformationen bisher nur die Spezifikation einer einfachen Deadline, bis zu der eine Aktivität beendet sein muss. Abhängigkeiten zwischen den Ausführungszeitpunkten der Aktivitäten können bisher genauso wenig berücksichtigt werden wie die Dauern der Aktivitäten. Eskalation ist somit erst nach Überschreiten der Deadline möglich, auch wenn dies bereits vorher absehbar wäre.

Eine schematische Übersicht der Architektur von ADEPT2 (inklusive des neuen Time-Managers) gibt Abbildung 7.2. ADEPT2 ist als eine Schichten-Architekur angelegt, welche eine strikte Trennung der einzelnen Bereiche erlaubt. Der Low-Level-Layer kümmert sich

mithilfe eines Datenbanksystems um die persistente Speicherung aller anfallenden Daten und die Verwaltung der verschiedenen Einstellungen. Der Basic-Services-Layer ist für die Speicherung und das Locking der verschiedenen Objekte (z. B. Prozessinstanzen) verantwortlich. Der Execution-Layer umfasst Funktionalitäten, wie die Prozess-Ausführung und das Änderungs-Management. Der User-Interaction-Layer stellt die Verbindung zum Benutzer her, wozu beispielsweise das Management der Arbeitslisten und die Benutzerschnittstelle gehören.

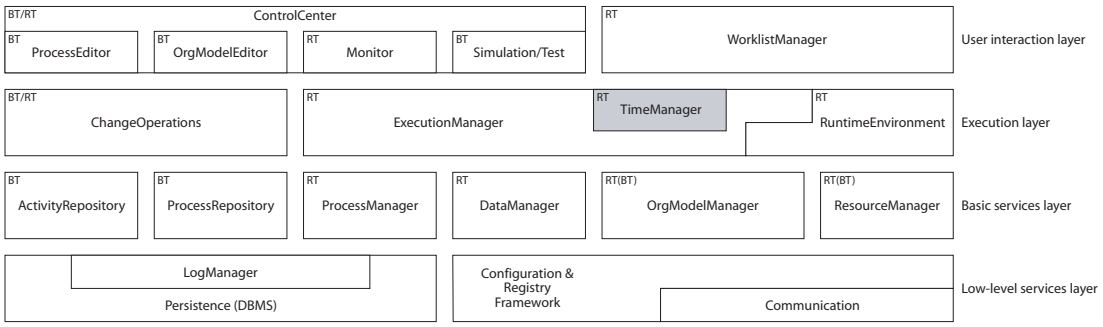



Abbildung 7.2: Architektur von ADEPT2 mit TimeManager (Übersicht) [Reichert *et al.*, 2008]

Die zentralen Komponenten der Prozessausführung sind der Prozess-Manager, der Data-Manager, der Execution-Manager, das Runtime-Environment und der Worklist-Manager. Der Prozess-Manager verwaltet die Prozessschemata und -instanzen. Der Data-Manager kümmert sich um die Speicherung und erneute Bereitstellung der während der Ausführung der Prozessinstanzen erzeugten Daten. Wird eine neue Prozessinstanz gestartet, interpretiert der Execution-Manager das zugehörige Prozessschema und ist für die Weiterschaltung anhand des Prozessgraphen zuständig. Die Ausführung der Aktivitäten sowie sämtliche damit verbundenen Schritte sind Aufgabe des Runtime-Environments. Der Worklist-Manager stellt die Arbeitslisten für die Clients bereit und kümmert sich um deren Aktualisierung.

## 7.2 Erweiterung der ADEPT2-API

Als zentrale Anlaufstelle für das Zeitmanagement in der erweiterten ADEPT2-API dient der Time-Manager, welcher in der Schicht-Architektur des ADEPT2-Systems neben dem Execution- und Runtime-Manager im Execution-Layer eingeordnet wird (siehe Abbildung 7.2). Die Schnittstelle des Time-Managers ist in Abbildung 7.3 zu sehen. Der parametrisierbare Typ (<T extends TimeSpan>) (siehe Anhang C) erlaubt ein einfaches

Austauschen der für die Zeitrepräsentation verwendeten Struktur (z. B. Intervalle oder Zeit-Histogramme etc.).

```
1  public interface TimeManager<T extends TimeSpan>
2  {
3    TimeUpdateManager<T> getTimeUpdateManager();
4  }
```

Abbildung 7.3: Time Manager

Zur Abfrage der zu den Prozessinstanzen gehörenden Zeitdaten stellt der Time-Manager den Time-Update-Manager bereit (siehe Abbildung 7.4). Bei diesem kann man sich für eine Benachrichtigung bei Änderungen des Zustandes von Aktivitäten registrieren (add/removeStateChangeListener). Daraufhin erhält man bei jeder Änderung des Zustandes einer Aktivität, egal ob Ausführungs- oder Zeitzustand, eine Benachrichtigung (inkl. des neuen Zustandes) über den bereitgestellten StateChangeListener. Die EBPInstanceReference erlaubt hierbei eine eindeutige Identifikation der betreffenden Aktivität, inklusive Instanz, Knoten und der aktuellen Iterationsnummer des Knotens (außerhalb von Schleifen immer 1). SessionToken dienen zur Identifikation des aktuellen Benutzers und erlauben darüber beispielsweise eine Zugriffskontrolle.

```
1  public interface TimeUpdateManager<T extends TimeSpan>
2  {
3    public interface StateChangeListener
4    {
5      void activityStateChanged(EBPInstanceReference activity,
6          NodeState nodeState, TimeState timeState);
7    }
8
9    void addStateChangeListener(StateChangeListener listener);
10
11   void removeStateChangeListener(StateChangeListener listener);
12
13   TimeDataAccess<T> getTimeDataAccess();
14
15   TimeState getTimeStateForActivity(SessionToken session,
16       EBPInstanceReference activity);
17
18   TimeState predictTimeStateForActivity(SessionToken session,
19       EBPInstanceReference activity, long timeStamp);
20 }
```

Abbildung 7.4: Time Update Manager

Daneben kann man vom Time-Update-Manager den aktuellen Zeitzustand einer Aktivität erfragen (getTimeStateForActivity) oder sich eine Vorhersage für den Zeitzustand zu

einem gewissen Zeitpunkt (unter der Annahme, dass bis dahin keine Änderung des Ausführungszustandes der Aktivität stattfindet) geben lassen (predictTimeStateForActvity).

Der vom Time-Update-Manager ebenfalls bereitgestellte Time-Data-Access (siehe Abbildung 7.5) erlaubt zusätzlich einen read-only Zugriff auf das für eine Prozessinstanz gültige Zeitmodell (getTimeModel), dessen minimales Netz (getMinimalNetwork) und den für die Prozessinstanz aktuell gültigen Schedule (getSchedule). Dabei dient eine UUID (Universal Unique Identifier) zur eindeutigen Identifikation der Prozessinstanz. Die Interfaces TemplateTimeModel und Schedule, welche man hierbei erhält, sind in Anhang C zu finden.

```
1  public interface TimeDataAccess<T extends TimeSpan>
2  {
3    TemplateTimeModel<T> getTimeModel(SessionToken session, UUID instanceID);
4
5    TemplateTimeModel<T> getMinimalNetwork(SessionToken session, UUID
         instanceID);
6
7    Schedule<T> getSchedule(SessionToken session, UUID instanceID);
8  }
```

Abbildung 7.5: Time Data Access

Um das Zeitmodell einer Prozessinstanz aktuell halten zu können, muss der Time-Manager über Änderungen des Ausführungszustandes der Aktivitäten der Prozessinstanz informiert werden. Hierzu dient das TimeNotification Interface (siehe Abbildung 7.6), über welches sich der Time-Manager bei seinem Start beim Execution-Manager registriert, um fortan über sämtliche Änderungen des Ausführungszustandes von Aktivitäten (activityStateChange) sowie das Starten und Beenden von Prozessinstanzen (instanceStarted, instanceFinished) informiert zu werden. Dabei erhält der Time-Manager zusätzlich den Zeitpunkt, zu dem das Ereignis eingetreten ist, um nicht von Verzögerungen bei der Bearbeitung und bei der Kommunikation zwischen den einzelnen Komponenten beeinflusst zu werden.

```
1  public interface TimeNotification
2  {
3    void instanceStarted(UUID instanceID, long timeStamp);
4
5    void instanceFinished(UUID instanceID, long timeStamp);
6
7    void activityStateChanged(EBPInstanceReference ebpInstanceReference,
8        NodeState state, long timeStamp);
9  }
```

Abbildung 7.6: Time Notification

Tritt ein nicht automatisch vom Time-Manager zu behandelnder Zeitfehler auf, wird das EscalationHandling-Interface (siehe Abbildung 7.7) benötigt. Es ist möglich, bei jeder Aktivität und jedem Prozessschema eine entsprechende Implementierung dieses Interface zu hinterlegen. Die Idee dahinter ist, dass Plug-ins (wie man sie vom Browser kennt) dem System zur Verfügung gestellt werden und dann eine eindeutige Plug-in-ID bei der Aktivität bzw. dem Prozessschema hinterlegt wird. Tritt ein Zeitfehler auf, wird entweder die bei der Aktivität hinterlegte Implementierung oder, sollte diese nicht gegeben sein, die des Prozessschemas aufgerufen, um den Zeitfehler zu korrigieren. Das Escalation-Handling bekommt hierzu zunächst sowohl die betroffene Instanz als auch deren aktuelles Zeitmodell mitgeteilt. Zusätzlich erhält es die Historie der Änderung (modificationHistory), die zur Entdeckung der Inkonsistenz geführt hat. Über diese ist es möglich, die inkonsistente Stelle des Zeitmodells zu lokalisieren (siehe Abschnitt 4.2.4.1). Die EscalationMeasures erlauben es dem Escalation-Handling das Zeitmodell zu korrigieren oder die Prozessinstanz abzubrechen.

```
1 public interface EscalationHandling<T extends TimeSpan>
2 {
3   void handleInconsistency(SessionToken session, UUID instanceID,
4       TemplateTimeModel<T> model, Modification<T> modificationHistory,
5       EscalationMeasures<T> measures);
6 }
```

Abbildung 7.7: Escalation Handling

Die Interaktionen der einzelnen Komponenten des erweiterten ADEPT2-Systems sind anhand einer beispielhaften Ausführung eines Prozesses in vereinfachter Form in Abbildung 7.8 zu finden. Da an dieser Stelle eine Trennung der verschiedenen Komponenten (Time-Update-Manager, Time-Notification, etc.) nicht sinnvoll ist, werden diese unter dem Time-Manager zusammengefasst. Der dargestellte Anwendungsfall beinhaltet zunächst das Starten einer neuen Prozessinstanz durch einen Client. Daraufhin steht die initiale Aktivität der Prozessinstanz zur Ausführung zur Verfügung und wird ebenfalls von einem Client gestartet. Nach dem Beenden der ersten Aktivität wird deren Nachfolger aktiviert und ebenfalls zur Ausführung bereitgestellt. Im vorgestellten Anwendungsfall kommt es durch Verzögerungen im Ablauf zu einer Verspätung, in deren Folge der eingetretene Zeitfehler durch das Escalation-Handling korrigiert werden muss.

### 7.2.1 Details der Implementierung

Die Zeitrechnung der Implementierung basiert, wie im Rahmen dieser Arbeit angenommen, auf Intervallen. Es ist jedoch ohne große Komplikationen möglich, diese beispielsweise gegen Zeit-Histogramme auszutauschen. Die Zeitbedingungen eines Prozessschemas werden, um die Änderungen am Prozess-Metamodell so gering wie möglich zu halten,

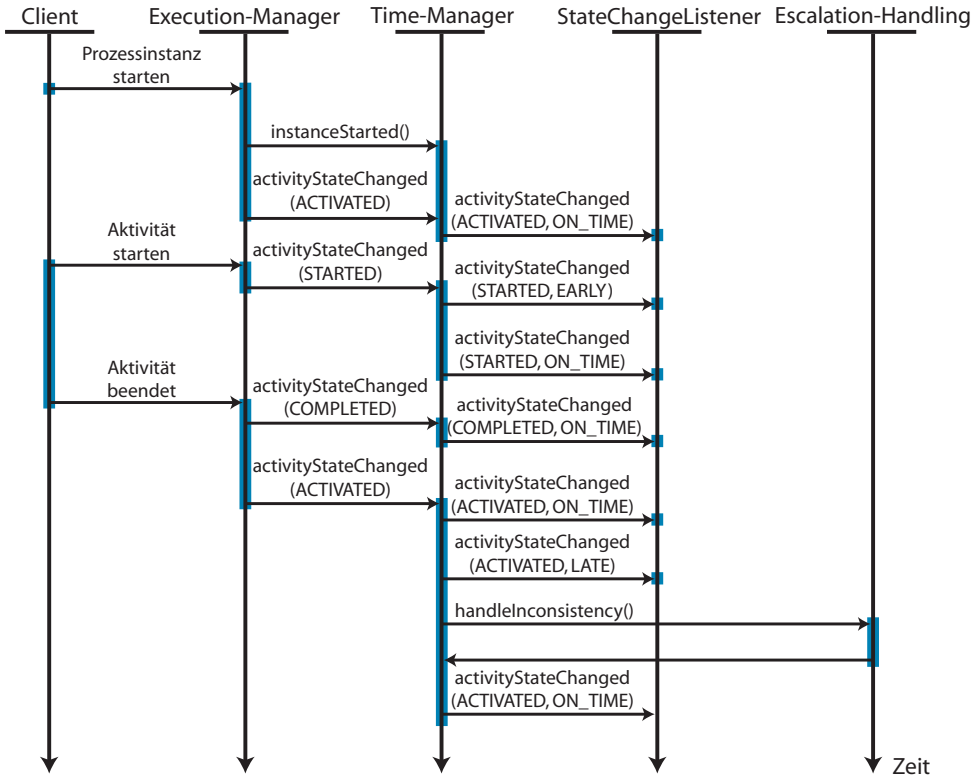



Abbildung 7.8: Interaktionsdiagramm einer beispielhaften Prozessausführung mit Time-Manager (vereinfacht)

hauptsächlich in den Meta-Informationen der Knoten und Kanten gespeichert. Eine Unterstützung unterschiedlicher Zeiteinheiten ist dabei bisher nicht vorhanden, es ist jedoch möglich, diese, mithilfe der in Abschnitt 4.2.2 angestellten Überlegungen, zu integrieren. Außerdem stellt dies keine wirkliche Einschränkung dar, da wie in Abschnitt 4.2.2 gezeigt, auch entsprechend angepasste Werte für die Zeitbedingungen verwendet werden können.

Der Time-Manager verwendet intern eine Time-Engine-Komponente zur Aktualisierung der Zeitdaten der Prozessinstanzen. Diese erlaubt das Setzen der Start- und End-Zeitpunkte der Aktivitäten sowie von deren Dauern. Für die Behandlung von Alternativ-Verzweigungen wird der in Abschnitt 4.2.5.1 diskutierte pessimistische Ansatz verwendet. Er wurde gewählt, weil er die beste Performance bietet. Bei der ermittelten Menge der Schedules handelt es sich um die in Abschnitt 4.2.4.2 besprochene Menge der freien Schedules.

Für die Speicherung der Zeitmodelle, minimalen Netze und Schedules steht eine Speicher-Schicht zur Verfügung, welche an das konkret verwendete System zur Speicherung der Daten (meist ein DBMS) angepasst werden kann. Die bisherige Implementierung ist jedoch nur transient, das heißt, die Daten werden lediglich im Hauptspeicher gespeichert und gehen somit bei einem Neustart des Systems verloren. Die Speicher-Schicht ist auch dafür verantwortlich, die gleichzeitige Veränderung der gespeicherten Elemente durch verschiedene Threads mittels Locking zu verhindern.

### 7.2.2 Einschränkungen der prototypischen Implementierung

Die bisherige Implementierung besitzt aus verschiedenen Gründen einige Einschränkungen, welche im Folgenden kurz erläutert werden. Zunächst einmal besteht eine zentrale Beschränkung darin, dass der Execution-Manager bisher keine Unterstützung für das TimeNotification Interface bietet. Da eine Implementierung dieser Unterstützung im Rahmen der Arbeit zu aufwendig gewesen wäre, wurde das Problem durch einen „Workaround“ gelöst. Anstatt die Time-Notification direkt als solche beim Execution-Manager zu registrieren, wurde sie in eine Worklist-Notification verpackt, welche normalerweise zur Benachrichtigung des Worklist-Managers über Änderungen an den zur Ausführung bereitstehenden Aktivitäten dient. Leider bietet diese keine Information über das Starten und Beenden von Instanzen. Diese muss daher aus der Information über die Ausführungszustands-Änderungen der Aktivitäten gewonnen werden musste. Auch bietet die Worklist-Notification keinerlei Zeitinformationen, weshalb hierfür der Zeitpunkt des Aufrufs der entsprechenden Methode verwendet wird, wodurch sich leichte Verzögerungen ergeben können.

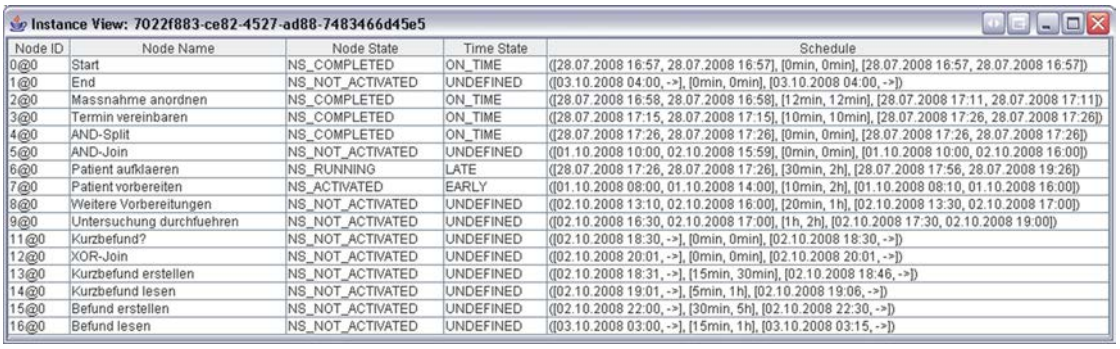
Ein Eskalations-Handling wurde bisher nicht implementiert, da es nicht Teil dieser Arbeit ist. Stattdessen wird die Zeitrechnung beim Auftreten eines Zeitfehlers abgebrochen und die Prozessinstanz ohne Zeitrechnung weiter fortgesetzt. Jedoch sind die nötigen Schnittstellen soweit implementiert, dass ein entsprechendes Eskalations-Handling jederzeit nachgereicht werden könnte.

Eine dritte Einschränkung ist, dass der graphische Editor zur Prozess-Modellierung bisher keine Modellierung von Zeitbedingungen unterstützt. Diese müssen daher mühsam von Hand in die Prozess-Beschreibung eingefügt werden, was jedoch ohne Weiteres möglich ist.

## 7.3 Benutzerschnittstelle

Zur Demonstration der Implementierung wurde eine rudimentäre Benutzerschnittstelle implementiert (siehe Abbildung 7.9). Diese erlaubt nach der Auswahl der gewünschten Prozessinstanz die Anzeige der Ausführungs- und Zeitzustände von deren Aktivitäten. Daneben wird der zu einer Aktivität gehörende Eintrag des aktuellen Schedules angezeigt, aus welchem die aktuellen Zeitbedingungen der Aktivität entnommen werden können.

Findet eine Änderung einer der beiden Zustände statt, wird die Ansicht, inklusive der aktuellen Schedules, entsprechend aktualisiert.



Instance View: 7022f883-ce82-4527-ad88-7483466d45e5

| Node ID | Node Name | Node State | Time State | Schedule |
|---|---|---|---|---|
| 0@0 | Start | NS_COMPLETED | ON_TIME | ([28.07.2008 16:57, 28.07.2008 16:57], [0min, 0min], [28.07.2008 16:57, 28.07.2008 16:57]) |
| 1@0 | End | NS_NOT_ACTIVATED | UNDEFINED | ([03.10.2008 04:00, ->], [0min, 0min], [03.10.2008 04:00, ->]) |
| 2@0 | Massnahme anordnen | NS_COMPLETED | ON_TIME | ([28.07.2008 16:58, 28.07.2008 16:58], [12min, 12min], [28.07.2008 17:11, 28.07.2008 17:11]) |
| 3@0 | Termin vereinbaren | NS_COMPLETED | ON_TIME | ([28.07.2008 17:15, 28.07.2008 17:15], [10min, 10min], [28.07.2008 17:26, 28.07.2008 17:26]) |
| 4@0 | AND-Split | NS_COMPLETED | ON_TIME | ([28.07.2008 17:26, 28.07.2008 17:26], [0min, 0min], [28.07.2008 17:26, 28.07.2008 17:26]) |
| 5@0 | AND-Join | NS_NOT_ACTIVATED | UNDEFINED | ([01.10.2008 10:00, 02.10.2008 15:59], [0min, 0min], [01.10.2008 10:00, 02.10.2008 16:00]) |
| 6@0 | Patient aufklaeren | NS_RUNNING | LATE | ([28.07.2008 17:26, 28.07.2008 17:26], [30min, 2h], [28.07.2008 17:56, 28.07.2008 19:26]) |
| 7@0 | Patient vorbereiten | NS_ACTIVATED | EARLY | ([01.10.2008 08:00, 01.10.2008 14:00], [10min, 2h], [01.10.2008 08:10, 01.10.2008 16:00]) |
| 8@0 | Weitere Vorbereitungen | NS_NOT_ACTIVATED | UNDEFINED | ([02.10.2008 13:10, 02.10.2008 16:00], [20min, 1h], [02.10.2008 13:30, 02.10.2008 17:00]) |
| 9@0 | Untersuchung durchfuehren | NS_NOT_ACTIVATED | UNDEFINED | ([02.10.2008 16:30, 02.10.2008 17:00], [1h, 2h], [02.10.2008 17:30, 02.10.2008 19:00]) |
| 11@0 | Kurzbefund? | NS_NOT_ACTIVATED | UNDEFINED | ([02.10.2008 18:30, ->], [0min, 0min], [02.10.2008 18:30, ->]) |
| 12@0 | XOR-Join | NS_NOT_ACTIVATED | UNDEFINED | ([02.10.2008 20:01, ->], [0min, 0min], [02.10.2008 20:01, ->]) |
| 13@0 | Kurzbefund erstellen | NS_NOT_ACTIVATED | UNDEFINED | ([02.10.2008 18:31, ->], [15min, 30min], [02.10.2008 18:46, ->]) |
| 14@0 | Kurzbefund lesen | NS_NOT_ACTIVATED | UNDEFINED | ([02.10.2008 19:01, ->], [5min, 1h], [02.10.2008 19:06, ->]) |
| 15@0 | Befund erstellen | NS_NOT_ACTIVATED | UNDEFINED | ([02.10.2008 22:00, ->], [30min, 5h], [02.10.2008 22:30, ->]) |
| 16@0 | Befund lesen | NS_NOT_ACTIVATED | UNDEFINED | ([03.10.2008 03:00, ->], [15min, 1h], [03.10.2008 03:15, ->]) |

Abbildung 7.9: Benutzerschnittstelle

Mithilfe dieser einfachen Benutzerschnittstelle ist es als Anwender bereits möglich, sich über die Zeitbedingungen seiner aktuellen Aktivitäten zu informieren. Es ist jedoch eine weitere Integration in den Client, speziell in dessen Anzeige der Arbeitslisten, wünschenswert. Dies ist aufgrund der bereitgestellten Schnittstellen ohne Weiteres möglich, liegt jedoch nicht im Rahmen dieser Arbeit.

## 7.4 Fazit

Wie die Implementierung zeigt, ist es möglich, ein Prozess-Management-System mit einer Zeitmanagementkomponente zu kombinieren. Dabei ist die vorgestellte Implementierung zunächst nur in der Lage, einen kleinen Teil der Funktionalitäten eines ausgewachsenen Zeitmanagements zu bieten. Jedoch hat sie einiges Potenzial, im Laufe der Zeit zu einem solchen erweitert zu werden.

Besonders die Wahl des ADEPT2-Modells und damit des ADEPT2-Systems hat sich bei der Implementierung bezahlt gemacht. Aufgrund der klar getrennten Struktur des ADEPT2-Systems und der sauber definierten und ausführlich dokumentierten Schnittstellen war es ohne Weiteres möglich, dieses um die gewünschte Funktionalität zu erweitern. Zudem konnte hierdurch weiterhin eine klare Trennung der einzelnen Komponenten erreicht werden, welche sich speziell bei weiteren Erweiterungen als vorteilhaft erweisen dürfte.

Eine Integration der Zeitaspekte wäre vor allem für den graphischen Editor wünschenswert. Bisher ist es zwar möglich, die Prozessschemata von Hand zu erweitern, dies ist jedoch nur für fortgeschrittene Benutzer empfehlenswert. Die nächsten Schritte sollten eine engere Integration des Time-Managers, insbesondere eine direkte Unterstützung

durch den Execution-Manager und die Arbeitslisten sowie eine adäquate Behandlung von Zeitfehlern sein.

# 8 Zusammenfassung und Ausblick

## 8.1 Zusammenfassung

In dieser Arbeit haben wir untersucht, wie eine Zeitmanagementkomponente für ein adaptives Prozess-Management-System realisiert werden kann. Dazu war zunächst von Interesse, welche Anforderungen vonseiten des Prozess-Metamodells an das Zeitmanagement bestehen und welche Arten von Konstrukten für eine ausreichende Modellierung der Zeitbedingungen eines Prozesses nötig sind. Dabei stellte sich unter anderem heraus, dass Alternativ-Verzweigungen und Schleifen vom Zeitmanagement nicht einfach zu behandeln sind. Außerdem konnte festgestellt werden, dass für eine ausreichende Modellierung der Zeitbedingungen, neben einigen Zeiteigenschaften für bereits existierende Konstrukte, ein weiteres Konstrukt, sogenannte Zeitkanten, zur Modellierung zeitlicher Abstände in das Prozess-Metamodell eingefügt werden muss.

Nachdem die Anforderungen an das Zeitmanagement abgegrenzt waren, haben wir verschiedene aus der Literatur bekannte Ansätze auf ihre Eignung für das Zeitmanagement in Prozess-Management-Systemen untersucht. Neben Arbeiten aus unterschiedlichen Bereichen der Informatik (z. B. Temporal Constraint Networks) wurden auch einige Ansätze aus anderen Gebieten, wie beispielsweise dem Prozessmanagement, betrachtet. Um einen besseren Überblick über das Thema Zeitmanagement zu erhalten, wurden darüber hinaus weitere Arbeiten, die nur am Rande mit dem Kern des Zeitmanagements zu tun haben, auf ihr Potenzial überprüft. Insgesamt stellte sich heraus, dass für das Zeitmanagement in adaptiven Prozess-Management-System einige weitergehende Anforderungen berücksichtigt werden müssen, die von den aktuellen Ansätzen nicht ausreichend abgedeckt werden.

Als Prozess-Management-System haben wir beispielhaft das ADEPT2-System genauer gesagt dessen Prozess-Metamodell, im nächsten Schritt um die benötigten Zeit-Eigenschaften sowie die zusätzlichen Zeitkanten zum ADEPT2-Time-Modell erweitert. Hierbei war für die ausreichende Unterstützung der Ausführung von Prozessinstanzen auch eine Erweiterung des Aktivitäten-Zustandes um einen Zeitzustand nötig. Dieser gibt anhand definierter Kriterien an, inwiefern die Prozessinstanz und deren Aktivitäten in der Lage sind, ihre Zeitbedingungen einzuhalten. Da das ADEPT2-Modell Alternativ-Verzweigungen und Schleifen unterstützt, haben wir zusätzlich betrachtet, welche Möglichkeiten es für deren Unterstützung gibt.

Im Anschluss wurde nach einem Konzept zur Darstellung und Behandlung von zeitlichen Einschränkungen gesucht und in den *Simple Temporal Problems* aus der Familie der

*Temporal Constraint Networks* gefunden. Diese bieten viele Vorteile, die ihren Einsatz zur Darstellung der Zeitbedingungen rechtfertigen. Zunächst haben wir damit ein Zeitmodell angegeben, welches auf Simple Temporal Problems beruht und eine einfache Transformation eines Prozessschemas in ein solches erlaubt. Auf Basis dieses Zeitmodells konnten Algorithmen entwickelt werden, welche die zeitliche Konsistenz des Zeitmodells und damit des Prozessschemas sicherstellen und Zeitgrenzen für die Aktivitäten der Prozessschemata bestimmen. Dabei wurden auch zwei der zuvor diskutierten Möglichkeiten zur Wiedergabe von Alternativ-Verzweigungen und Schleifen auf die verwendeten Algorithmen übertragen. Zuletzt wurde erörtert, wie im Fall von zu restriktiven Zeitbedingungen mittels einer Abschwächung dieser eventuell Abhilfe geschaffen werden kann.

Aus den verwendeten Zeiteigenschaften an sich und auch aus den gewählten Verfahren ergaben sich für die Modellierung der zeitlichen Ebene eines Prozesses einige neue Aspekte, wie beispielsweise eine Beziehung zwischen den Dauern von Aktivitäten und den Abständen zwischen diesen, die anhand eines Beispiel-Prozesses erörtert wurden.

Auf Grundlage der modellierten Prozessschemata folgt die Ausführung ihrer Instanzen. Hier stand die Betrachtung im Vordergrund, wie das Zeitmodell einer Prozessinstanz entsprechend der realen Ausführungszeitpunkte der Aktivitäten aktualisiert und die Einhaltung der spezifizierten Zeitbedingungen überwacht werden können. Daneben haben wir untersucht, wie mit speziellen Zeitbedingungen verfahren werden kann, welche erst während der Laufzeit bekannt sind. All dies diente dem Ziel, eventuelle Zeitfehler vermeiden oder unmittelbar auf diese reagieren zu können. Lässt sich ein Zeitfehler nicht vermeiden, muss eine spezielle Ausnahmebehandlung stattfinden, um den Fehler zu korrigieren. Hierfür wurden verschiedene grundlegende Möglichkeiten vorgestellt, wie dies geschehen kann. Teil eines adaptiven Prozess-Management-Systems sind auch dynamische Modifikationen der Prozessschemata während deren Ausführung. Daher haben wir ermittelt, welche dieser Modifikationen vom Zeitmanagement betroffen sind und welche zusätzlichen Einschränkungen sich hierdurch ergeben.

Ein interessantes Anwendungsgebiet für das Zeitmanagement ist die Prognose von Zeitdaten. Aus diesem Grund wurde ermittelt, wie diese auf Basis der ermittelten Zeitdaten erfolgen kann. Im Interesse stand dabei auch, welchen Nutzen beispielsweise ein Ressourcenmanagement oder die persönliche Terminplanung der Benutzer des Systems aus diesen Prognosen ziehen kann.

Ein Teil der Diplomarbeit bildete auch die prototypische Implementierung zentraler Komponenten in einem konkreten System, um deren Realisierbarkeit zu demonstrieren. Es hat sich dadurch bereits gezeigt, dass es sowohl möglich ist, eine Zeitmanagementkomponente in ein adaptives Prozess-Management-System zu integrieren als auch, dass die vorgestellten Konzepte und Algorithmen auf konkrete Abläufe anwendbar sind.

Wie wir gezeigt haben, machen die beschriebenen Anforderungen eine umfassende Erweiterung heutiger Systeme notwendig. Eine ausreichende Unterstützung zeitlicher Aspekte wird für die weitere Verbreitung von Prozess-Management-System jedoch eine wichtige Rolle spielen. Die vorliegende Arbeit konnte einen Weg zur Realisierung einer

Zeitmanagementkomponente für ein adaptives Prozess-Management-System aufzeigen. Jedoch ist auch deutlich geworden, dass das Ende des Weges noch nicht erreicht ist.

## 8.2 Ausblick

Die in dieser Arbeit vorgestellte Möglichkeit zur Realisierung einer Zeitmanagementkomponente ist ein weiterer Schritt auf dem Weg der Prozess-Management-Systeme zu einem intelligenten Assistenzsystem. Um dieses Ziel zu verwirklichen, sind aber noch einige weitergehende Fragestellungen zu untersuchen.

Einige Bereiche des Zeitmanagements, wie beispielsweise eine ausreichende Behandlung von Zeitfehlern, wurden im Rahmen dieser Arbeit aufgrund des begrenzten Zeitrahmens ausgeklammert. Jedoch haben wir für viele dieser Fragestellungen entsprechende Ansatzpunkte aufgezeigt, von denen eine weitere Entwicklung ausgehen kann.

Ein weiterer Punkt ist, die Übertragung der verbliebenen Möglichkeiten zur Behandlung verschiedener Instanztypen auf das vorgestellte Zeitmodell und die zugehörigen Algorithmen. Insbesondere eine nähere Betrachtung des Ansatzes des kürzesten und längsten Pfades könnte hierbei von Interesse sein, da der bisher untersuchte pessimistische Ansatz einige korrekte Prozess-Modelle als inkonsistent einstuft. Auch die Entwicklung weiterer Ansätze zur Behandlung unterschiedlicher Instanztypen, die insbesondere eine bessere Unterstützung für Schleifen bieten, sind nicht auszuschließen.

Auch die Unterstützung unterschiedlicher Zeiteinheiten bietet noch zahlreiche interessante Fragestellungen. Die vorgestellte Umrechnung bietet eine rudimentäre Möglichkeit, um Zeiteinheiten behandeln zu können. Jedoch werden einige Fragen von dieser gar nicht oder nur unzureichend beachtet. Eine weitergehende Unterstützung könnte sicherlich dazu beitragen, die Modellierung und Ausführungssemantik der Zeitbeziehungen noch intuitiver zu gestalten.

Ein wichtiges Thema, welches in dieser Arbeit nur am Rande behandelt werden konnte, ist die Behandlung von Zeitfehlern. Bisher muss an dieser Stelle immer ein menschlicher Experte eingreifen, um das Problem zunächst vollständig zu lokalisieren und dieses dann zu beheben. Hier wäre eine weitergehende Unterstützung durch das System wünschenswert. Beispielsweise könnten für verschiedene wiederkehrende Situationen entsprechende Schemata hinterlegt werden, die eine automatische Behandlung des Zeitfehlers erlauben oder dem Experten verschiedene Vorschläge unterbreiten. Durch eine genauere Analyse des Zeitfehlers und dessen Ursachen könnte der Experte ebenfalls bei seiner Arbeit unterstützt werden. Eine vollständige Automatisierung der Fehlerbehandlung erscheint jedoch unrealistisch, da diese meist von zu vielen äußeren Umständen abhängt, die dem System unbekannt sein dürften.

Zur Vorbeugung von Zeitfehlern ist eine Erweiterung der eingeführten Zeitzustände dahin gehend wünschenswert, dass die an einer Prozessinstanz beteiligten Bearbeiter früher auf Verzögerungen im Betriebsablauf aufmerksam gemacht und hierdurch in die

Lage versetzt werden, entsprechend zu reagieren. Eng hiermit verbunden sind auch persönliche Terminpläne für jeden Bearbeiter, die diesem eine bessere Planung seines Arbeitsablaufes und damit eine Vermeidung von Zeitfehlern ermöglichen. Dabei kann auch ein Ressourcenmanagement mit in die Terminplanung einfließen, welches auf Basis der berechneten Zeitdaten realisiert werden kann und noch genauere Terminpläne erlauben würde.

Ein weiteres Thema ist die Behandlung von Schleifen. In diesem Zusammenhang gibt es, über die bereits besprochenen Aspekte hinaus, noch sehr viele offene Fragestellungen, wie beispielsweise der Umgang mit Schüsselaktivitäten innerhalb von Schleifen oder auch die Ressourcenzuteilung für regelmäßig wiederkehrende Aktivitäten.

Abschließend lässt sich feststellen, dass das Prozess-Management ein sehr intuitives und mächtiges Konzept ist, welches in der Lage ist, den Benutzer von lästiger Routinearbeit zu befreien und dabei gleichzeitig Fehlerquellen reduzieren kann. Die technischen Möglichkeiten eines Prozess-Management-Systems werden in den nächsten Jahren sicherlich, nicht zuletzt um eine Unterstützung zeitlicher Aspekte, weiter wachsen und somit zu einer weiteren Verbreitung solcher Systeme führen.

# Literatur


Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Pages 832–843 of: Communications of the ACM.* ACM.

Alur, R., & Dill, D. 1994. A Theory of Timed Automata. *Theoretical Computer Science*, **126**(2), 183–235.

Attie, P., Singh, M., Sheth, A., & Rusinkiewicz, M. 1993. Specifying and Enforcing Intertask Dependencies. *In: Proceedings of 19th VLDB.*

Bause, F., & Kritzinger, P. 1998. Stochastic Perti Nets: An Introduction to the Theory. *ACM SIGMETRICS Performance Evaluation Review*, **26**(2), 2–3.

Bellmann, R. 1958. On a routing problem. *Quart. Appl. Math*, **16**, 87–90.

Bettini, C., Wang, X. S., & Jajodia, S. 1998. A general framework for time granularity and its application to temporal reasoning. *Annals of Methematics and Artifical Intelligence*, **22**, 29–58.

Bettini, C., Wang, X. S., & Jajodia, S. 2000. Free Schedules for Free Agents in Workflow Systems. *In: 7th International Workshop on Temporal Representation and Reasoning (TIME2000).*

Bettini, C., Wang, X. S., & Jajodia, S. 2002a. Solving multi-granularity temporal constraint networks. *Artif. Intell.*, **140**(1/2), 107–152.

Bettini, C., Wang, X. S., & Jajodia, S. 2002b. Temporal Reasoning in Workflow Systems. *Chap. 11, pages 269–306 of: Distributed and Parallel Databases.*

Braig, E. 2004. *Unterstützung von Zeitaspekten in Workflow-Management-Systemen.* Diplomarbeit, Universität Ulm.

Chen, J., & Yang, Y. 2004. Temporal Dependency for Dynamic Verification of Temporal Constraints in Workflow Systems. *Proc. of the 3 rdInternational Conference on Grid and Cooperative Computing, Springer-Verlag, LNCS*, **3251**, 1005–1008.

Chen, J., & Yang, Y. 2005a. An Activity Completion Duration based Checkpoint Selection Strategy for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems. *Pages 296–310 of: Proc. of 2 ndInternational Conference on Grid Service Engineering and Management (GSEM2005).*

Chen, J., & Yang, Y. 2005b. *Multiple Consistency States of Fixed-time Constraints in Grid Workflow Systems.* Tech. rept. Technical Report, Faculty of ICT, Swinburne University of Technology, Mar. 2005, http://www. it. swin. edu. au/personal/yyang/papers/2005TR-Jchen-1. pdf.

Chen, J., & Yang, Y. 2007. Adaptive selection of necessary and sufficient checkpoints for dynamic verification of temporal constraints in grid workflow systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, **2**.

Chen, J., Yang, Y., & Chen, T. 2004. Dynamic verification of temporal constraints on-the-fly for workflow systems. *Pages 30–37 of: Proc. 11th Asia-Pacific Software Engineering Conference.*

Combi, C., Franceschet, M., & Peron, A. 2004. Representing and Reasoning about Temporal Granularities. *Journal of Logic and Computation*, **14**(1), 51–77.

Dadam, P., & Reichert, M. 1998. The ADEPT WfMS Project at the University of Ulm. *In: Proceedings of the 1 st European Workshop on Workflow and Process Management (WPM'98) (Workflow Management Research Projects).*

Dadam, P., Kuhn, K., Reichert, M., Beuter, T., & Nathe, M. 1995. ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen. *Proc. 25. GI Jahrestagung und 13. Schweizer Informatikertag*, 677–685.

Dadam, P., Reichert, M., & Kuhn, K. 2000. Clinical Workflows - The Killer Application for Process-oriented Information Systems. *Pages 36–59 of: 4th Internaltional Conference on Business Information Systems (BIS' 2000).*

Dadam, P., & Reichert, M. 2004. ADEPT - Prozess-Management-Technologie der nächsten Generation. *Pages 27–43 of: Aktuelle Trends in der Softwareforschung.* IRB Verlag Stuttgart.

Dadam, P., Reichert, M., Rinderle, S., Jurisch, M., Acker, H., Goeser, K., Kreher, U., & Lauer, M. 2008. Towards Truly Flexible and Adaptive Process-Aware Information Systems. *In: UNISCON 2008.*

Dechter, R., Meiri, I., & Pearl, J. 1991. Temporal constraint networks. *Artifical Intelligence*, **49**, 61–95.

DIN. 1987. *DIN 69 900 Teil 1 und 2: Netzplantechnik.*

Eder, J., & Panagos, E. 2000. *Managing Time in Workflow Systems.* Future Strategies INC. Pages 109–132.

Eder, J., & Pichler, H. 2002. Duration Histograms for Workflow Systems. *Pages 239–253 of: Proceedings of the IFIP TC8 / WG8.1 Working Conference on Engineering Information Systems in the Internet Context*, vol. 231.

Eder, J., Panagos, E., & Rabinovich, M. 1999a. Time Constraints in Workflow Systems. *Pages 286–300 of:* Jarke, M., & Oberweis, A. (eds), *CAiSE*. Lecture Notes in Computer Science, vol. 1626. Springer.

Eder, J., Euthimios, P., Pozewaunig, H., & Rabinovich, M. 1999b. Time Management in Workflow Systems. *Pages 265–280 of: 3d International Conference on Business Information Systems*.

Eder, J., Gruber, W., & Panagos, E. 2000. Temporal Modeling of Workflows with Conditional Execution Paths. *Pages 243–253 of:* Ibrahim, M. T., Küng, J., & Revell, N. (eds), *DEXA*. Lecture Notes in Computer Science, vol. 1873. Springer.

Eder, J., Pichler, H., Gruber, W., & Ninaus, M. 2003. Personal Schedules for Workflow Systems. *Pages 216–231 of:* van der Aalst, W. M. P., ter Hofstede, A. H. M., & Weske, M. (eds), *International Conference on Business Process Management*. Lecture Notes in Computer Science, vol. 2678. Springer.

Eder, J., Bierbaumer, M., & Pichler, H. 2005. Calculation of Delay Times for Workflows with Fixed-Date Constraints. *Pages 544–547 of: CEC*. IEEE Computer Society.

Eder, J., Eichner, H., & Pichler, H. 2006. A Probabilistic Approach to Reduce the Number of Deadline Violations and the Tardiness of Workflows. *Pages 5–7 of:* Meersman, R., Tari, Z., & Herrero, P. (eds), *OTM Workshops (1)*. Lecture Notes in Computer Science, vol. 4277. Springer.

Ellis, C., & Nutt, G. 1993. Modeling and Enactment of Workflow Systems. *In: Application and Theory of Petri Nets 1993: 14th International Conference, Chicago, Illinois, USA, June 21-25*. Springer.

Evans, J., & Edward, M. 1992. *Optimization Algorithms for Networks and Graphs*. M.Dekker.

Georgakopoulos, D., Hornick, M., & Sheth, A. 1995. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, **3**, 119–153.

Goralwalla, I., Leontiev, Y., Özsu, M., Szafron, D., & Combi, C. 2001. Temporal Granularity: Completing the Puzzle. *Journal of Intelligent Information Systems*, **16**(1), 41–63.

Grimm, M. 1997. *ADEPT-TIME: Temporale Aspekte in flexiblen Workflow-Management-Systemen.* Diplomarbeit, Universität Ulm.

Heinlein, C. 2001. Workflow and Process Synchronization with Interaction Expression and Graphs. *Ulmer Informatik Berichte*, **2000-11**.

Hensinger, C. 1997. *ADEPT-Flex Dynamische Modifikation von Workflows und Ausnahmebehandlung in WfMS.* Diplomarbeit, Universität Ulm, Fakultät für Informatik.

Hensinger, C., Reichert, M., Bauer, T., Strzeletz, T., & Dadam, P. 2000. ADEPTworkflow - Advanced Workflow Technology for the Efficient Support of Adaptive, Enterprise-wide Processes. *In: Software Demonstration Track, Held in conjunction with EDBT '00 conference.*

Holliday, M. A., & Vernon, M. K. 1985. A Generalized Timed Petri Net Model for Performance Analysis. *Pages 181–190 of: International Workshop on Timed Petri Nets.* Washington, DC, USA: IEEE Computer Society.

Jurisch, M. 2006. *Konzeption eines Rahmenwerks zur Erstellung und Modifikation von Prozessvorlagen und -instanzen.* Diplomarbeit, Universität Ulm.

Kafeza, E., & Karlapalem, K. 2000. Gaining Control over Time in Workflow Management Applications. *Page 232 of: Database and Expert Systems Applications: 11th International Conference, DEXA 2000, London, UK, September 2000.* Lecture Notes in Computer Science, vol. 1973/2000. Springer Berlin / Heidelberg.

Kecher, C. 2006. *UML 2.0 - Das umfassende Handbuch.* Galileo Computing.

Kerbosh, J., & Schell, H. 1975. Network Planning by the Extended METRA Potential Method. *Report KS-1.1, University of Technology, Eindhoven, Department of Industrial Engineering.*

Konyen, I., Schultheiß, B., & Reichert, M. 1996. *Prozeßentwurf für den Ablauf einer radiologischen Untersuchung.* Tech. rept. Abteilung Datenbanken und Informationssysteme, Universität Ulm.

Li, H., & Yang, Y. 2005. Dynamic checking of temporal constraints for concurrent workflows. *Electronic Commerce Research and Applications*, **4**, 124–142.

Li, J., Fan, Y., & Zhou, M. 2003. Timing constraint workflow nets for workflow analysis. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, **33**(2), 179–193.

Luttman, R., Laffel, G., & Pearson, S. 1995. Using PERT/CPM (Program Evaluation and Review Technique/Critical Path Method) to design and improve clinical processes. *Qual Manag Health Care*, **3**(2), 1–13.

Ly, L. T., Göser, K., Rinderle-Ma, S., & Dadam, P. 2008. Compliance of Semantic Constraints - A Requirements Analysis for Process Management Systems. *In: Proc. 1st Int'l Workshop on Governance, Risk and Compliance - Applications in Information Systems (GRCIS'08).*

Mackworth, A. 1975. Consistency in Networks of Relations. *Artifical Itelligence*, **8**, 99–118.

Maria, E. D., Montanari, A., & Zantoni, M. 2006. An automaton-based approach to the verification of timed workflow schemas. *Pages 87–94 of: TIME.* IEEE Computer Society.

Marjanovic, O., & Orlowska, M. E. 1999. On Modeling and Verification of Temporal Constraints in Production Workflows. *Knowl. Inf. Syst.*, **1**(2), 157–192.

Neumann, K., & Morlock, M. 1993. *Operations Research.* Carl Hanser Verlag.

Oestereich, B. 1997. *Objektorientierte Softwareentwicklung.* Oldenbourg R. Verlag.

(OMG), O. M. G. 2006. *Unified Modeling Language: Infrastructure.* http://www.omg.org/docs/formal/07-02-04.pdf.

Panagos, E., & Rabinovich, M. 1997. Predictive Workflow Management. *In:* Silberschatz, A., & Shoval, P. (eds), *NGITS.*

Papadimitriou, C. H. 1994. *Computational Complexity.* Addison Wesley.

Petri, C. 1962. Kommunikation mit Automaten. *Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM*, **2**, 65–377.

Pozewaunig, H. 1996. *Behandlung von Zeit in Workflow-Managementsystemen – Modellierung und Integration.* Diplomarbeit, Universität Klagenfurt.

Pozewaunig, H., Eder, J., & Liebhart, W. 1997. ePert: Extending PERT for Workflow Managment Systems. *Pages 217–224 of: Advances in Databases and Information Systems.*

Reichert, M., & Dadam, P. 1997. A Framework for Dynamic Changes in Workflow Management Systems. *DEXA Workshop*, 42–48.

Reichert, M., & Dadam, P. 1998. Adept flex: Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, **10**(2), 93–129.

Reichert, M., Dadam, P., & Bauer, T. 2003. Dealing with Forward and Backward Jumps in Workflow Management Systems. *Journal Software and Systems Modeling (SOSYM)*, **2**, 37–58.

Reichert, M., Rinderle, S., Kreher, U., & Dadam, P. 2005. Adaptive Process Management with ADEPT2. *In: Int'l Conf. on Data Engineering, ICDE '05.*

Reichert, M. 2000. *Dynamische Ablaufänderungen in Workflow-Management-Systemen.* Dissertation, Universität Ulm.

Reichert, M., & Bauer, T. 2007. Supporting Ad-hoc Changes in Distributed Workflow Management Systems. *In: 15th Int'l Conf. on Coop. Information Systems (CoopIS'07).*

Reichert, M., Dadam, P., Jurisch, M., Kreher, U., Goeser, K., & Lauer, M. 2008. Architectural Design of Flexible Process Management Technology. *In: Proc. PRIMIUM Subconference at the Multikonferenz Wirtschaftsinformatik (MKWI).*

Rinderle, S., Reichert, M., & Dadam, P. 2004a. On Dealing with Structural Conflicts between Process Type and Instance Changes. *In: Business Process Management: Second International Conference, BPM 2004.* Springer.

Rinderle, S., Reichert, M., & Dadam, P. 2003. Evaluation of Correctness Criteria for Dynamic Workflow Changes. *In: Proc. 1st Int'l Conf. on Business Process Management (BPM '03).*

Rinderle, S., Reichert, M., & Dadam, P. 2004b. Correctness Criteria for Dynamic Changes in Workflow Systems: A Survey. *Data & Knowledge Engineering*, **50**, 9–34.

Rinderle, S., Reichert, M., & Dadam, P. 2004c. Flexible Support of Team Processes by Adaptive Workflow Systems. *Distributed and Parallel Databases*, **16**, 91–116.

Rinderle, S., Kreher, U., Lauer, M., Dadam, P., & Reichert, M. 2006. On Representing Instance Changes in Adaptive Process Management Systems. *In: Proc. First IEEE Workshop on Flexibility in Process-aware Information Systems (ProFlex'06).*

Rinderle-Ma, S., Reichert, M., & Weber, B. 2008. Relaxed Compliance Notions in Adaptive Process Management Systems. *In: Proceedings 27th Int'l Conference on Conceptual Modeling (ER'08).*

Rinza, P. 1994. *Projektmanagement: Planung, Überwachung und Steuerung von technischen und nichttechnischen Vorhaben.* VDI-Verlag.

Sadiq, W., Marjanovic, O., & Orlowska, M. E. 2000. Managing Change and Time in Dynamic Workflow Processes. *Int. J. Cooperative Inf. Syst.*, **9**(1-2), 93–116.

Sapolsky, H. 1972. *The Polaris system development; bureaucratic and programmatic success in government.* Harvard University Press.

Saynisch, M., Schelle, H., & Schub, A. 1979. *Projektmanagement: Konzepte, Verfahren, Anwendungen.* Oldenbourg R. Verlag GmbH.

Schwalb, E., & Dechter, R. 1997. Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, **93**(1-2), 29–61.

Shaffer, L., Ritter, J., & Meyer, W. 1965. *The Critical-path Method.* McGraw-Hill Education.

Stucky, W. 1995. *Automaten Sprachen Berechenbarkeit.* Teubner Verlag, Stuttgart.

Tsai, J., Yang, J., *et al.* 1995. Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications. *IEEE Transactions on Software Engineering*, **21**(1), 32–49.

Tsang, E. 1993. *Foundations of constraint satisfaction.* Academic Press San Diego.

van der Aalst, W., & Weijters, A. 2005. Process Mining. *Process-Aware Information Systems: Bridging People and Software Through Process Technology.*

van der Aalst, W., van Hee, K., & Houben, G. 1994. Modelling and analysing workflow using a Petri-net based approach. *Pages 31–50 of:* Michelis, G., Ellis, C., & Memmi, G. (eds), *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms.*

van der Aalst, W., De Michelis, G., & Ellis, C. 1998. Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98). *In: Proc. Lisbon, Portugal, Eindhoven University of Technology, Eindhoven, The Netherlands, Computing Science Report*, vol. 98.

van der Aalst, W., Weske, M., & Grünbauer, D. 2005. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, **53**(2), 129–162.

Weber, B., Rinderle, S., & Reichert, M. 2007. Change Patterns and Change Support Features in Process-Aware Information Systems. *In: Int'l Conf. on Advanced Information Systems Engineering, CAiSE 2007.*

WfMC. 1995. *The Workflow Reference Model.*

WfMC. 1999 (Feb). *The Workflow Management Coalition - Terminology & Glossary.* http://www.wfmc.org.

Wikipedia. 2008. *Mittelwert — Wikipedia, Die freie Enzyklopädie.* [http://de.wikipedia.org/w/index.php?title=Mittelwert&oldid=48771978; Stand 18. August 2008].

Wodtke, D., & Weikum, G. 1997. A Formal Foundation for Distributed Workflow Execution Based on State Charts. *In: Database Theory–ICDT'97: 6th International Conference, Delphi, Greece, January 8-10, 1997: Proceedings.* Springer.

Zerguini, L. 2001. On the Estimation of the Response Time of the Business Process. *In: 17th UK Performance Engineering Workshop.*

Zhao, J. L., & Stohr, E. A. 1999. Temporal Workflow Management in a Claim Handling System. *Pages 187–195 of: Proc. Int'l Joint Conf. on Work Activities Coordination and Collaboration (WACC'99)*, vol. 24. ACM SIGSOFT Notes.

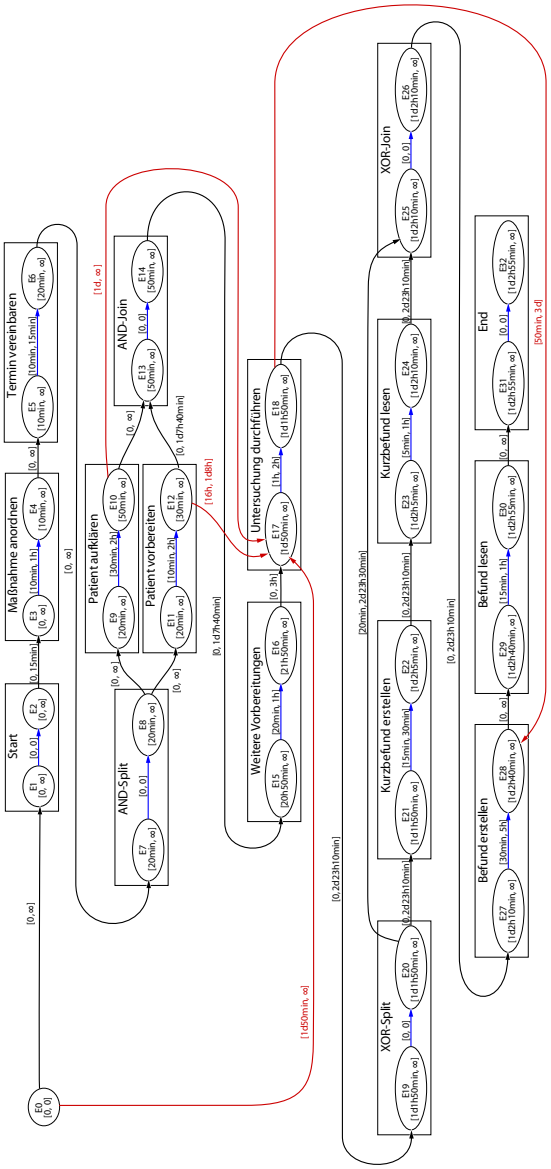
Zhuge, H., Cheung, T.-Y., & Pung, H.-K. 2001. A timed workflow process model. *Pages 231–243 of: Journal of Systems and Software*, vol. 55.

# A Notation (in alphabetischer Reihenfolge)

| Symbol | Bedeutung |
|---|---|
| $\prec$ | Vorgängerrelation (Seite 21) |
| $\preccurlyeq$ | Transitive Vorgängerrelation (Seite 21) |
| $\succ$ | Nachfolgerrelation (Seite 21) |
| $\succcurlyeq$ | Transitive Nachfolgerrelation (Seite 21) |
| $[M]$ | Intervall von Werten der Menge $M$ (Seite 58) |
| $\mathbf{A}$ | Menge der Kanten eines Zeitmodells (Seite 76) |
| $Activity\_n$ | Menge der Knoten mit einer Aktivität (Seite 19) |
| $c_{ij}$ | Constraint zwischen den Variablen $i$ und $j$ (Seite 76) |
| $Control\_e$ | Menge der Kontrollkanten (Seite 19) |
| $\mathbf{D}$ | Menge der Datenelemente eines Prozesses (Seite 18) |
| $D^{min}$ | Minimale Dauer einer Aktivität (Seite 26) |
| $D^{max}$ | Maximale Dauer einer Aktivität (Seite 26) |
| $\mathbf{DF}$ | Menge der Datenkanten eines Prozesses (Seite 18) |
| $e = (\mathtt{ID}^{start}, \mathtt{ID}^{end}, \mathtt{eT}, \mathtt{RP}^{start}, \mathtt{RP}^{end}, \mathtt{Time})$ | Kante eines Prozessschemas (Seite 19 und 61) |
| $\mathbf{E}$ | Menge der Kanten eines Prozesses (Seite 18) |
| $\mathbf{eT}$ | Menge der Kantentypen (Seite 19) |
| $FZ^{event}$ | Frühester Zeitpunkt des Ereignisses *event* (Seite 58) |
| $\mathbf{\Gamma}$ | Menge der Constraints eines Zeitmodells (Seite 76) |
| $Join\_n$ | Menge der Join-Knoten (Seite 19) |
| $Loop\_e$ | Menge der Schleifenrückwärtskanten (Seite 19) |
| $n = (\mathtt{ID}, \mathtt{nT}, \mathbf{Pars}, \mathbf{M}, \mathtt{Time})$ | Knoten eines Prozessschemas (Seite 18 und 59) |
| $\mathbf{N}$ | Menge der Knoten eines Prozesses (Seite 18) |
| $\mathbf{NG}$ | Menge der Ausführungszustände (Seite 22) |
| $\mathbf{nT}$ | Menge der Knotentypen (Seite 18) |
| $P = (\mathbf{N}, \mathbf{E}, \mathbf{D}, \mathbf{DF})$ | Prozess (Seite 18) |
| $Split\_n$ | Menge der Split-Knoten (Seite 19) |
| $Sync\_e$ | Menge der Sync-Kanten (Seite 19) |
| $SZ^{event}$ | Spätester Zeitpunkt des Ereignisses *event* (Seite 58) |
| $\mathbf{TD}$ | Menge der möglichen Dauern (Seite 58) |

| Symbol | Bedeutung |
|---|---|
| $TM = (\mathbf{V}, \mathbf{A}, \mathbf{\Gamma})$ | Zeitmodell (Seite 76) |
| $\mathbf{TP}_{Cal}$ | Menge der möglichen Zeitpunkte eines Kalenders (Seite 58) |
| $\mathbf{TP}_{NP}$ | Menge der möglichen Zeitpunkte mit Bezugspunkt $NP$ (Seite 58) |
| $\mathbf{TP}_{Ws}$ | Menge der möglichen Zeitpunkte relativ zum Start des Workflow (Seite 58) |
| $\mathbf{V}$ | Menge der Knoten eines Zeitmodells (Seite 76) |
| $Z^{min}$ | Minimaler Abstand zweier Aktivitäten (Seite 30) |
| $Z^{max}$ | Maximaler Abstand zweier Aktivitäten (Seite 30) |

# B Zeitmodell des Beispielprozess



# C ADEPT2-Time-Schnittstellen

### Time Span

```
1  package adept2time.model.timemodel;
2
3  /**
4   * A Time Span represents a period of time in some form e.g. a interval or a
5   * duration histogram.
6   *
7   * @author Andreas Lanz
8   */
9  public interface TimeSpan extends Comparable<TimeSpan>
10 {
11
12   /**
13    * Non-destructively calculates the composition of this time span and the
14    * given time span.
15    *
16    * @param duration The Time Span the composition is calculated with
17    * @return The combined period of the of the two time spans
18    */
19   TimeSpan composition(TimeSpan duration);
20
21   /**
22    * Non-destructively calculates the intersection of this time span and the
23    * give time span
24    *
25    * @param duration The Time Span the intersection is calculated against.
26    * @return The Intersection of the two periods of time
27    */
28   TimeSpan intersect(TimeSpan duration);
29
30   /**
31    * Returns the negation of this Time Span i.e. the Time Span with a
32    * negative sign.
33    *
34    * @return The negation of this Time Span
35    */
36   TimeSpan negate();
37
38   /**
39    * Returns whether this Time Span overlaps the other Time Span
40    *
41    * @param duration The Time Span which may overlap with this one
42    * @return Whether this Time Span overlaps <code>duration<code> or not.
43    */
44   boolean overlaps(TimeSpan duration);
45
46   /**
```

```
47      * Returns whether this Time Span contains the other Time Span
48      *
49      * @param duration The Time Span which may be contained within this one
50      * @return Whether this Time Span contains <code>duration<code> or not.
51      */
52    boolean contains(TimeSpan duration);
53
54    /**
55      * Returns whether this period of time is empty or not.
56      *
57      * @return Whether this period of time is empty or not.
58      */
59    boolean isEmpty();
60
61  }
```

## Constraint

```
1   package adept2time.model.timemodel;
2
3   /**
4    * This interface represents a Constraint within a Time Model. A Constraints
5    * start at one Variable and End at another. It has an associated Time
6    * Restriction which represents the restrictions between the two variables
7    * made by this constraint.
8    * <p />
9    * The Constraint contains some Meta information like if it's implicit or
10   * explicit or if it's a duration constraint which are mainly for display
11   * purposes.
12   *
13   * @author Andreas Lanz
14   *
15   * @param <T> The concrete Class used for representing the Time Restriction
16   */
17  public interface Constraint<T extends TimeSpan> extends Cloneable
18  {
19
20    /**
21      * The ID of the Variable the Constraint start at
22      *
23      * @return The Variable this Constraint starts at
24      */
25    int getStartVariableID();
26
27    /**
28      * The ID of the Variable the Constraint ends at.
29      *
30      * @return The Variable this Constraint ends at
31      */
32    int getDestinationVariableID();
33
34    /**
35      * Non-destructively calculates the intersection between the two
36      * Constraints. That is a constraint with the same start and end Variables
37      * as this whose time Restriction is the Intersection of the time
38      * Restrictions of both constraints.
```

```
39      *
40      * @see TimeSpan#intersect(TimeSpan)
41      *
42      * @param other The Constraint the intersection is calculated against. It
43      *              must start and end at the same Variables as this constraint.
44      * @return A Constraint representing intersection of the two constraints
45      */
46     Constraint<T> intersect(Constraint<T> other);
47
48     /**
49      * Non-destructively calculates the composition of the two Constraints.
50      * That is a constraint with the same start variable as this and the same
51      * end variable as the other whose time Restriction is the composition of
52      * the time Restrictions of both constraints.
53      *
54      * @see TimeSpan#composition(TimeSpan)
55      *
56      * @param other The constraint the composition is calculated with. It must
57      *              start at the same variable this ends at.
58      * @return A constraint representing the composition of the two constraints
59      */
60     Constraint<T> composition(Constraint<T> other);
61
62     /**
63      * Returns the same Constraint but pointing into the other direction i.e.
64      * the start and destination variables are switch and the time Restriction
65      * is negated.
66      *
67      * @return A constraint representing the inversion of this.
68      */
69     Constraint<T> invert();
70
71     /**
72      * Returns whether this constraint is inconsistent i.e. it time
73      * Restriction is empty.
74      *
75      * @return Whether this constraint is inconsistent i.e. it time
76      *              Restriction isempty.
77      */
78     boolean isInconsistent();
79
80     /**
81      * The Time Restriction established by this Constraint.
82      *
83      * @return The Time Restriction established by this Constraint.
84      */
85     T getTimeRestriction();
86
87     /**
88      * Returns whether this constraint has been implicitly calculated during
89      * the updates of the time model or whether it has been explicitly given.
90      *
91      * @return Whether this Constraint has been explicitly given or not.
92      */
93     boolean isImplicit();
94
95     /**
96      * Returns whether this constraint has been generated be the inversion of
```

```
97      * another constraint or not.
98      *
99      * @return Whether this constraint has been generated be the inversion of
100     * another constraint or not.
101     */
102   boolean isNegated();
103
104   /**
105    * Whether this Constraint represents the duration of an activity or not.
106    * This is solely for Display purposes.
107    *
108    * @return Whether this Constraint represents the duration of an activity
109    *          or not.
110    */
111   boolean isDuration();
112
113   Constraint<T> clone();
114
115 }
```

### Time Model

```
1  package adept2time.model.timemodel;
2
3  import java.util.Set;
4
5  /**
6   * A Time Model represents a Simple Temporal Problem. It contains a set of
7   * variables and constraints connecting them.<br />
8   * A Time Model may be a minimal Network or not.
9   *
10  * @author Andreas Lanz
11  *
12  * @param <T> The concrete Class used for representing the Time Restriction
13  */
14 public interface TimeModel<T extends TimeSpan>
15 {
16   /**
17    * Returns the Constraint representing the restriction between the two
18    * variables.
19    *
20    * @param startVariableID The ID of the Variable the Constraint starts at.
21    * @param destVariableID The ID of the Variable the Constraint ends at.
22    * @return The Constraint between the two Variables
23    */
24   Constraint<T> getConstraint(int startVariableID, int destVariableID);
25
26   /**
27    * Checks whether there has been a Constraint set between the two Variable
28    * or not.
29    *
30    * @param startVariableID The ID of the Variable the Constraint starts at.
31    * @param destVariableID The ID of the Variable the Constraint ends at.
32    * @return <code>true</code> if there has been a constraint set,
33    *         <code>false</code> otherwise
34    */
```

```
35    boolean constraintSet(int startVariableID, int destVariableID);
36
37    /**
38     * Returns a List of all Variables within this Time Model
39     *
40     * @return A List of all Variables
41     */
42    Set<Integer> getVariables();
43
44    /**
45     * Returns whether this Time Model represents a Minimal Network or not.
46     *
47     * @return Returns whether this Time Model is minimal or not
48     */
49    boolean isMinimal();
50
51    ChangeableTimeModel<T> clone();
52  }
```

### Template Time Model

```
1   package adept2time.model.timemodel;
2
3   import java.util.Set;
4
5   import de.aristaflow.adept2.model.processmodel.Template;
6
7   /**
8    * A Template Time Model represents a Time Model which is associated with a
9    * Template i.e. the Time Models Variables represent the starting and ending
10   * of the templates activities.
11   *
12   * @author Andreas Lanz
13   *
14   * @param <T> The concrete Class used for representing the Time Restriction
15   */
16  public interface TemplateTimeModel<T extends TimeSpan> extends TimeModel<T>
17  {
18    /**
19     * Returns a set of all Activity Nodes which are in this model. This will
20     * be the node IDs of all Nodes in {@link Template#getNodes()}
21     *
22     * @return The ID's of all activity nodes in this model.
23     */
24    Set<NodeIterationReference> getActivities();
25
26    /**
27     * Retrieves the Duration Constraint of the Activity identified by it's
28     * node ID and iteration.
29     *
30     * @param activity The ID of the activities Node
31     * @param iteration The Iteration of the activities Node
32     * @return The Duration of the given Activity as set by the constraints of
33     *         this Time Model
34     */
35    Constraint<T> getActivityDuration(NodeIterationReference activity);
```

```
36
37    /**
38     * Retrieves the Start Time Constraint of the Activity identified by it's
39     * node ID and iteration.
40     *
41     * @param activity The ID of the activities Node
42     * @param iteration The Iteration of the activities Node
43     * @return The Start Time Constraint of the given Activity as set by the
44     *         constraints of this Time Model
45     */
46    Constraint<T> getActivityStartTime(NodeIterationReference activity);
47
48    /**
49     * Retrieves the End Time Constraint of the Activity identified by it's
50     * node ID and iteration.
51     *
52     * @param activity The ID of the activities Node
53     * @param iteration The Iteration of the activities Node
54     * @return The End Time Constraint of the given Activity as set by the
55     *         constraints of this Time Model
56     */
57    Constraint<T> getActivityEndTime(NodeIterationReference activity);
58
59    /**
60     *
61     * Returns the Pair of Variables (start- and end-Variable) which represent
62     * the activity in this model.
63     *
64     * @param activity The ID of the activities Node
65     * @param iteration The Iteration of the activities Node
66     *
67     * @return The pair of Variables (start- and end-Variable) which
68     *         represent the activity in this model
69     */
70    Tupel<Integer> getVariablesForActivity(NodeIterationReference activity);
71
72    /**
73     * Returns the ID of the activity whose starting or ending is represented
74     * by the given variable.
75     *
76     * @param variableID The ID of the Variable
77     * @return The Activity represented by the Variable
78     */
79    NodeIterationReference getActivityForVariable(int variableID);
80
81    /**
82     * Returns whether the constraint between the two variables has been
83     * retained e.g. after the activity whose duration is represented by the
84     * constraint has been executed and the duration is fixed.
85     *
86     * @param startVariableID The ID of the Variable the Constraint starts at.
87     * @param destVariableID The ID of the Variable the Constraint ends at.
88     * @return Whether the Constraint has been retained or not.
89     */
90    boolean isConstraintRetained(int startVariableID, int destVariableID);
91
92
93    /**
```

```
94      * Returns the dynamic Time Span Function which should be applied to the
95      * constraint between the two given variables to determine it's new time
96      * restriction.
97      *
98      * @param startVariableID The ID of the Variable the Constraint starts at
99      * @param destVariableID THe ID of the Variable the Constraint ends at
100     * @return The dynamic Time Span Function to be used to update the
101     *         constraint or <code>null</code> if there is none.
102     */
103   DynamicTimeSpanFunction<T> getDynamicTimeSpan(int startVariableID, int
        destVariableID);
104
105   /**
106    * Returns the dynamic Time Limit Function which should be applied to the
107    * start and end time of the given activity to determine there new values.
108    *
109    * @param activity The ID of the Activity the Time Limit Function applies
110    *        to
111    * @return The dynamic Time Limit Function to be used to update the
112    *         activities start and end time or <code>null</code> if there is
         none.
113    */
114   DynamicTimeLimitFunction<T> getTimeLimitFunction(NodeIterationReference
        activity);
115 }
```

## Schedule

```
1  package adept2time.model.timemodel;
2
3  import java.util.Set;
4
5  /**
6   * A Schedule is a set of triples for each activity. The first value is the
7   * possible start time of the activity, the second it's duration and the
8   * third it's possible end time.
9   *
10  * @author Andreas Lanz
11  *
12  * @param <T> The concrete Class used for representing Time
13  */
14 public interface Schedule<T extends TimeSpan>
15 {
16   /**
17    * The Activities which have time datas in this Schedule.
18    *
19    * @return A set of all Activities which have entries in this schedule.
20    */
21   Set<NodeIterationReference> getActivities();
22
23   /**
24    * The Start Time of the Activity
25    *
26    * @param activity The ID of the activity
27    * @return The possible Start Time of the Activity
28    */
```

```
29    T getScheduledStartTime(NodeIterationReference activity);
30
31    /**
32     * The Duration of the Activity
33     *
34     * @param activity The ID of the activity
35     * @return The possible Duration of the Activity
36     */
37    T getScheduledDuration(NodeIterationReference activity);
38
39    /**
40     * The End Time of the Activity
41     *
42     * @param activity The ID of the activity
43     * @return The possible End Time of the Activity
44     */
45    T getScheduledEndTime(NodeIterationReference activity);
46
47  }
```

# Index

# Glossar

### Historie

Eine Historie ist ein Ausführungsplan, welcher angibt zu welchen Zeitpunkten und in welcher Reihenfolge Operationen beziehungsweise Aktivität ausgeführt wurden. Sie wird bei PMS meist im Hintergrund erstellt, um beispielsweise eine Möglichkeit zu besitzen, die abgelaufenen Vorgänge nachträglich zu überprüfen.

### Metamodell

Ein Metamodell ist ein Modell zur Beschreibung von Modellen. Es definiert, die Syntax, d. h. die in den Modellen verfügbaren Konstrukte und die Regeln, nach denen aus diesen ein gültiges Modell zusammengesetzt werden kann, und die Semantik dieser Konstrukte. Ein Beispiel für ein Metamodell ist das UML-Metamodell [(OMG), 2006]

### Mittelwert

Ein Mittelwert ist ein Wert aus der Statistik, der die Lage der Werte einer Verteilung oder Stichprobe beschreibt. Ziel ist es dabei die wesentliche Information einer längeren Reihe von (Mess-)Daten in wenigen Daten zu konzentrieren. Es gibt verschiedene Mittelwerte, die nach unterschiedlichen Kriterien gebildet werden. Die bekanntesten sind arithmetisches, geometrische und harmonisches Mittel, Median und Modus [Wikipedia, 2008].

### NP-vollständig

NP-vollständig ist ein Begriff aus der Komplexitätstheorie für Algorithmen. Er bezeichnet eine Reihe von Problemen, für welche keine effizienten Lösungsalgorithmen bekannt sind [Papadimitriou, 1994]. Die bekanntesten NP-vollständigen Probleme sind das Traveling-Salesman-Problem, das Erfüllbarkeitsproblem der Aussagenlogik und das Rucksackproblem.

### Organisationsmodell

Das Organisationsmodell ist ein Modell, welches die Organisationseinheiten und ihre Beziehungen darstellt. Es beschreibt Konzepte der Hierarchie, der Befugnis und der Verantwortlichkeiten der Einheiten. Zusätzlich kann es eine Reihe von Attributen

enthalten, welche die Einheiten näher beschreiben, beispielsweise Fähigkeiten und Rollen.

### Prozess-Reengineering

Beim Prozess-Reengineering werden die bisherigen Prozesse einer Organisation unter verschiedenen Gesichtspunkten wie Effizienz, Durchlaufzeit, Fehlerquoten, etc. überarbeitet.

### Semantik

Die Semantik beschreibt die Bedeutung einer Aussage. Dabei wird die Bedeutung einer Aussage durch die Semantik nicht erforscht, sondern explizit durch Regeln festgelegt. Sie kann dazu genutzt werden eine durch sie beschriebene Aussage zu interpretieren.