ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für
Ingenieurwissenschaften
und Informatik
Institut für Datenbanken
und Informationssysteme

Yasemin Agbulak
yasemin.agbulak@uni-ulm.de

2011

Adaptation und Definition von Prozessmodellen auf Basis von Concurrent Task Trees

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Yasemin Agbulak

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Jens Kolb

Fassung 31. Mai 2011

Satz: PDF-LaTeX 2ε

Zusammenfassung Software-entwickelnde Unternehmen sind heutzutage gezwungen, Entwicklungsaufwände zu reduzieren und dabei ständig steigende Bedürfnisse der Benutzer zu erfüllen. Vielfach misslingt es aufgrund mangelnder Ressourcen und speziellen Know-Hows, diesen Anforderungen in ausreichendem Maße zu genügen. Die Folgen davon sind erfahrungsgemäß schlecht gestaltete Benutzeroberflächen, die ein kostenintensives Nacharbeiten nach sich ziehen. Um diesen Folgen entgegenzuwirken und somit erfolgreiche Benutzerschnittstellen zu entwerfen, ist eine enge Zusammenarbeit von Endbenutzern als Experten für die Bedienoberfläche und Entwicklern als Experten für die technischen Aspekte einer Anwendung erforderlich. Probleme bereitet allerdings häufig die praktische Realisierung von Beteiligung der Benutzer. Graphische Methoden stellen hier eine intuitive und einfache Alternative zu komplexen textuellen Beschreibungen dar, womit die Integration der Endbenutzer in den Entwicklungsprozess erreicht werden soll. In dieser Arbeit untersuchen wir die Concurrent Task Trees (CTT) als Endbenutzer-orientierte Modellierungssprache für Prozessmodelle. Dazu werden wir die CTT-Notation mit einer weiteren Modellierungssprache ADEPT vergleichen, um gewisse Ähnlichkeiten oder Unterschiede festzustellen und daraus schließend die Vor- und Nachteile der CTT-Notation für diesen Einsatzzweck zu ermitteln.

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren haben die Entwickler von Benutzeroberflächen erkannt, dass die Integration von Endnutzern und deren Anforderungen in den Entwicklungsprozess ein sehr wichtiger Faktor für den Erfolg des zu entwickelnden Software-Produktes ist. Übliche Entwicklungsprozesse weisen auf, dass der Schwerpunkt beim Interface-Design nicht auf gebrauchstauglicher Seite, sondern eher auf funktionaler Seite liegt. Dies ist eines der Hauptgründe, weshalb herkömmliche Entwicklungsprozesse für die Entwicklung benutzerorientierter Software nur bedingt geeignet sind [1]. Basis eines nutzerzentrierten Entwicklungsprozesses ist die Fokussierung auf die eventuell zukünftigen, bisherigen oder potenziellen Nutzer sowie deren Aufgaben, Vorgehensweisen und Arbeitsbedingungen. Für die Umsetzung und Einbringung dieser Aspekte in den Softwareentwicklungsprozess, ist es von großer Notwendigkeit die zukünftigen Endbenutzer bereits in die Entwurfsphase des Systems zu involvieren. Da noch kein vollständiger Entwicklungsprozess existiert, der eine Integration zwischen Nutzerzentrierung und klassischer Softwareentwicklung vorweisen kann, besteht die Möglichkeit, diese zwei Aspekte durch die Benutzung formaler Modelle zu verbinden. In der Softwareentwicklung ist die Modellierung der Applikationslogik, z.B. mit UML [2], gängige Praxis. Solche Modellierungssprachen gelten als ein wichtiges Mittel, um die kommunikative Kluft zwischen Software-Fachleuten und Endbenutzern zu überbrücken [3].
Auch werden Geschäftsprozessmodelle in der Regel von Personen erstellt, die einem der Informatik fremden Umfeld zuzuordnen sind. Notationen zur Modellierung von Geschäftsprozessen sollten somit den technischen Aspekt der späteren Umsetzung vernachlässigen, damit die Prozessmodelle leicht verständlich sind und als Diskussionsgrundlage dienen können. Jedoch erschweren größe Prozessmodelle, trotz ausdrucksstarker Prozessbeschreibungssprachen, die Verständlichkeit und erhöhen die Komplexität.
In dieser Arbeit werden wir die *Concurrent Task Trees* (CTT) auf die Prozessbeschreibungssprache *ADEPT* anwenden, um zu sehen, welche Ergebnisse bzw. Verbesserungen bzgl. Prozessmodellierung erzielt werden können.
Paterno [4] entwickelte mit der Sprache CTT eine derartige Notation für das Design von Benutzerschnittstellen. Im CTT liegt der Fokus auf den Benutzern und ihren Aufgaben. Sie eignet sich für die Spezifizierung und Modellierung von Benut-

zeraufgaben. Durch ihre hierarchische Baumstruktur ermöglicht CTT verschiedene task models, wie z.B. Benutzeraufgaben und Systemaufgaben, zu spezifizieren und zu modellieren. Die graphische Syntax des CTT ist einfach zu erlernen und zu benutzen, mit der Endbenutzer ihre persönlichen Wünsche und Anforderungen an das zu entwickelnde System ausdrücken können. Im Zusammenspiel mit den definierten Systemaufgaben seitens der Entwickler werden diese CTT-Modelle später in konkrete Anwendungen umgesetzt werden.
Um zunächst wichtige Voraussetzungen für die Betrachtungen in dieser Arbeit sicherzustellen, wird in Kapitel 2 ausführlich über die Grundlagen von CTT gesprochen. Dabei werden die unterschiedlichen Aufgabentypen, die verschiedenen temporalen Relationen und die Umgebung für die Modellierung von CTTs vorgestellt.
Das Kapitel 3 beschäftigt sich mit dem Ändern von Prozessmodellen. Dabei werden sogenannte *Change Pattern* auf CTT-Modelle und ADEPT-Modelle angewandt und die daraus resultierenden Ergebnisse diskutiert. Im Besonderen werden Ähnlichkeiten und Unterschiede der bereits genannten Notationen herangezogen und erörtert.
In Kapitel 4 werden verschiedene mögliche Ausprägungen von CTT-Modellen anhand von Fallstudien untersucht und bewertet.
Kapitel 5 fasst schließlich die Erkenntnisse dieser Arbeit und die Vorteile der vorgestellten Modellierungssprache CTT zusammen und gibt einen kurzen Ausblick über weitere interessante Themen.

2 Modellierung von Prozessen in CTT

In diesem Kapitel werden sowohl die wesentlichen Grundlagen des CTT, als auch die Identifikation von Zielen und notwendigen Aufgaben, sowie ihre Verfeinerung in kleinere Teilaufgaben, eingeführt und vorgestellt.
Das Vorgehen bei der Modellierung von Aufgaben beginnt damit, einzelne Aufgaben und Teilaufgaben zu identifizieren und zueinander in Beziehung zu setzen. Dabei liegt der Schwerpunkt auf den Aufgaben, die lediglich von Benutzern, mit oder ohne Interaktion mit der Anwendung, ausgeführt werden. In diesem Zusammenhang bietet CTT fünf verschiedene Typen von Aufgaben (s. Abschn. 2.1). Die hierarchische Strukturierung von Aufgaben ist ein grundlegender Aspekt in der Modellierung mittels CTT. Die Definition der Beziehungen zwischen Aufgaben kann allerdings sehr komlpex werden, so dass dafür eine Werkzeugunterstützung existiert (s. Abschn. 2.4).
Zuvor werden aber noch zentrale Merkmale von CTT aufgeführt [5], die in den folgenden Abschnitten und Kapiteln von Bedeutung sind:

1. Hierarchische Struktur: Mit der baumartigen Struktur bietet CTT eine Möglichkeit, Aufgaben auf einer sehr intuitiven Art und Weise hierarchisch zu strukturieren und zu modellieren. Dies ist nah an der menschlichen Art und Weise Aufgaben zu lösen. Beim Problemlösen tendiert der Mensch dazu, das zu lösende Problem in kleinere Teilprobleme bzw. Teilaufgaben zu zerlegen und zu lösen, und hierbei die Beziehungen zwischen den einzelnen Teilaufgaben und Teillösungen nicht aus den Augen zu verlieren.

2. Grafische Syntax: Eine grafische Syntax ist sowohl für Endbenutzer als auch für Experten häufig leichter zu interpretieren; aus Punkt 1 folgt, dass im Fall der CTT die logische Struktur einer graphischen Notation eine baumartige Form haben muss.

3. Temporale Operatoren: Die Operatoren des CTT (s. Abschn. 2.2) ermöglichen es, den Informationsfluss innerhalb der Aufgaben einer Hierarchieebene zu beschreiben. Die Beziehungen zwischen den Aufgaben werden in Form von Richtungslinien dargestellt, die die Richtung des Informationsflusses und die zeitliche Ausführungsreihenfolge der Aufgaben angeben.

4. Fokus auf Aktivitäten: CTT stellt die Aktivitäten, die die Nutzer durchzuführen haben, in den Vordergrund und versucht von Implementierungsdetails zu abstrahieren. Dieser Blickwinkel erlaubt es Entwicklern, sich auf die relevantesten Aspekte bei der Gestaltung interaktiver Anwendungen zu konzentrieren. Diese umfassen sowohl benutzer- als auch systembezogene Aspekte. Dadurch wird die Betrachtung technischer Implementierungsdetails vermieden, die während der Entwurfsphase zu Verwirrungen führen und wichtige Faktoren des Entwurfs in den Schatten stellen würden.

2.1 Spezifizierung der Aufgaben

Bei der graphischen CTT-Notation werden die Aufgaben in fünf verschiedene Typen eingeteilt. Eine Aufgabe bzw. *Task* definiert, wie der Benutzer in einer bestimmten Anwendung ein Ziel erreichen kann. In diesem Zusammenhang ist das Ziel eine gewünschte Änderung des Systemzustands oder eine Anfrage an das System.
Zum einen gibt es die *Usertasks*, die vollständig durch den Anwender ausgeführt werden ohne Beteiligung der Anwendung (s. Abb. 2.1 a). Sie erfordern kognitive oder körperliche Aktivitäten zur Entscheidungsfindung. Ein mögliches Beispiel eines Usertasks ist das Auswählen eines Urlaubsorts in einer Web-Anwendung.
Applicationtasks werden ausschließlich seitens des Systems durchgeführt. Sie erhalten Informationen aus dem System und können ebenso Informationen an den Benutzer liefern (s. Abb 2.1 b). Applicationtasks werden durch die Anwendung aktiviert. Beispiel für Applicationtasks sind das Kompilieren eines Programmcodes und das Senden von Exception-Nachrichten, wenn Fehler entdeckt werden, oder das Empfangen und Anzeigen von Netzwerknachrichten.
Interaktionen zwischen Anwender und System heißen *Interactiontasks* (s. Abb. 2.1 c). Diese Interaktionen werden durch den Anwender aktiviert. Als Beispiel einer solchen Interaktion ist das Formulieren einer Datenbankabfrage, deren Ausführung und die Ausgabe der entsprechenden Resultate.
Abstractiontasks sind alle Aufgaben, die noch nicht eindeutig zu den obigen drei Kategorien zugeordnet werden können (s. Abb. 2.1 d). Sie werden durch ihre untergeordneten Tasks beschrieben. Aufgaben, deren Unteraufgaben unterschiedlicher Typen sind, sind in CTT abstrakte Aufgaben.
Cooperationtasks sind Aufgaben, die Teilaufgaben besitzen (s. Abb. 2.1 e). Diese können durch mehrere Benutzer bearbeitet werden und kommen in kooperativen CTTs zum Tragen. Welche Benutzer zu welchen Aktionen berechtigt sind, wird über Rollen definiert.

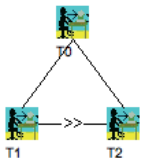
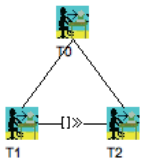
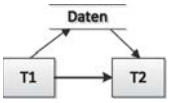
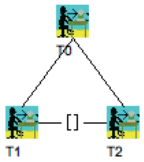
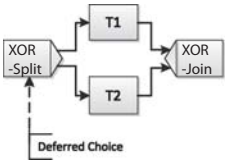
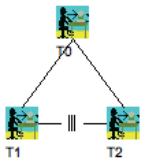
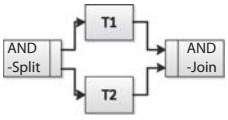
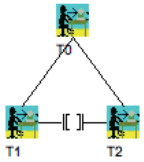
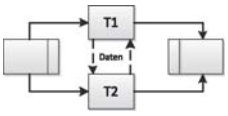
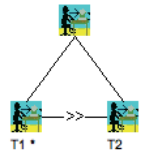
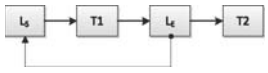
(a) Usertask (b) Application-task (c) Interaction-task (d) Abstraction-task (e) Cooperation-task

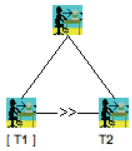
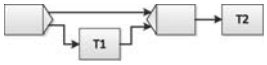
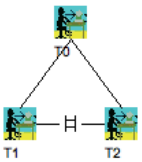
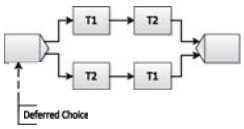
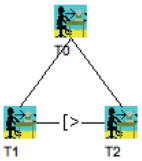
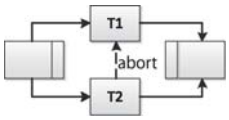
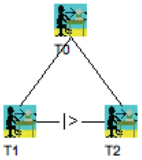
Abbildung 2.1: Komponenten des CTT

Die unterschiedlichen Aufgabentypen werden über figurative Symbole repräsentiert. Dies hat den Vorteil, dass die verschiedenen Aufgabentypen bei großen Aufgabenmodellen aus der Vogelperspektive einfacher erkannt werden als die Bezeichnungen der Aufgaben. Sie unterstützen damit den Modellierer beim Erfassen oder Navigieren der Aufgabenhierarchie.
Die Einordnung einer Aufgabe in einen Tasktypen hängt von den Tasktypen der Unteraufgaben im Baum ab. Wenn alle Unteraufgaben aus dem gleichen Typen bestehen, dann muss auch die Oberaufgabe zu diesem Tasktyp gehören. Wenn die Unteraufgaben aus unterschiedlichen Tasktypen stammen, dann muss die Oberaufgabe zum Abstractiontask zählen. Blattaufgaben dürfen nicht als Abstractiontask definiert werden.

2.2 Operatoren zur Beschreibung temporaler Relationen

In diesem Abschnitt werden die möglichen temporalen Relationen beschrieben, die die zeitliche Abfolge von Aufgaben definieren. Eine temporale Beziehung im CTT-Modell wird zwischen zwei Aufgaben derselben Hierarchieebene mit demselben Vaterelement durch Verbindungen mit Operatoren annotiert. Das heißt mit einem temporalen Operator wird nicht die Beziehung der Unteraufgaben zu einer Oberaufgabe beschrieben, sondern die Beziehung von zwei “benachbarten” Aufgaben. Die folgende Tabelle listet die verfügbaren Operatoren auf:

temporale Relation in CTT		Erläuterung
		**Enabling**: Aufgabe T2 wird erst durch Ausführung und Beendigung von Aufgabe T1 aktiviert.
		**Enabling with Info Exchange**: Stellt dieselbe zeitliche Abfolge wie Enabling dar, nur mit dem Unterschied, dass Informationen von T1 an T2 übermittelt werden.
		**Choice**: Hierbei handelt es sich um das sogenannte *Deferred Choice*. Sobald eine Wahl getroffen wird, sind die anderen Alternativen automatisch nicht mehr ausführbar.
		**Concurrent**: Die Aufgaben T1 und T2 werden in beliebiger Reihenfolge parallel ausgeführt.
		**Concurrent with Info Exchange**: T1 und T2 können gleichzeitig ausgeführt werden, allerdings müssen sie miteinander synchronisieren, um Informationen auszutauschen.
		**Iteration**: T1 wird wiederholt durchgeführt, bis es von einer anderen Aufgabe - in dem Fall T2 - deaktiviert wird.

temporale Relation in CTT		Erläuterung
		**Option**: Vor der Ausführung von T2 kann T1 ausgeführt werden oder auch nicht.
		**Order Independency**: Beide Aufgaben müssen ausgeführt werden. Allerdings kann die eine Aufgabe erst dann starten, wenn die andere mit der Durchführung fertig ist.
		**Disabling**: Sobald T2 begonnen hat, wird T1 abgebrochen und beendet.Z.B. die Aufgabe *durchlesen einer Webseite* und die Aufgabe *Auswahl eines Links*, welche auf eine andere Webseite führt.
		**Suspend** / **Resume**: T2 kann T1 unterbrechen. Wenn T2 abgeschlossen ist, kann T1 von dem vor der Unterbrechung erreichten Zustand aus weitergeführt werden.

2.3 Ablauf der Modellierung

Die Modellierung von CTT-Modellen richtet sich an die menschliche Art und Weise Aufgaben zu lösen. Der Mensch tendiert dazu, die zu lösende Aufgabe Schritt für Schritt in kleinere Teilaufgaben zu zerlegen und zu lösen, und hierbei die Beziehungen zwischen den einzelnen Teilaufgaben nicht aus den Augen zu verlieren. Als Ergebnis aus solch einer Zerlegung und Gliederung der Aufgaben entsteht eine hierarchische Baumstruktur. Somit orientiert sich die Vorgehensweise der Modellierung einem Top-Down-Ansatz, bei dem man vom Abstrakten, Allgemeinen, Übergeordneten schrittweise hin zum Konkreten, Speziellen, Untergeordneten geht. Wesentlich für die Aufgabenmodellierung mit Hilfe von CTT ist die Identi-

fikation von Zielen und notwendigen Aktivitäten sowie ihre Verfeinerung in kleinere Aufgaben. Im Folgenden sollen hier wichtige Begriffe abgegrenzt werden, die im weiteren Verlauf zum besseren Verständnis beitragen sollen. Da das CTT-Modell eine baumartige Struktur besitzt halten wir uns an die Terminologie von Bäumen:

- Baum → Gerichteter und zusammenhängender Graph, der aus einer Menge von Knoten und Kanten besteht
- Knoten → entspricht einem Task im CTT-Modell
- Vaterknoten → Vorgänger eines Knotens auf der darüberliegenden Ebene, im CTT-Modell immer genau einer
- Kindknoten → Nachfolger eines Knotens auf der darunterliegenden Ebene, im CTT-Modell mindestens zwei oder mehrere Kindknoten erlaubt
- Geschwisterknoten → Knoten mit gleichem Vater
- Kante → beschreibt die Verbindung zwischen zwei Knoten, jedoch nicht die Verbindung zwischen Vater- und Kindknoten; entspricht einer temporalen Relation im CTT-Modell
- Pfad → Folge unterschiedlicher, durch Kanten verbundener Knoten
- Wurzel → ausgezeichneter Knoten, der keine Vorgänger hat
- Blatt → Knoten ohne Nachfolger
- Teilbaum → Knoten mit all seinen direkten und indirekten Nachfolgern
- Linker Teilbaum → linker Kindknoten + Teilbaum, der daran hängt
- Rechter Teilbaum → rechter Kindknoten + Teilbaum, der daran hängt
- Ebene → Level eines Baums (s. Abb. 2.2)
- Blattebene → ist die Ebene, die am weitesten von der Wurzel entfernt ist

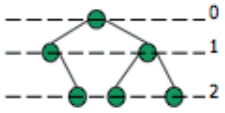
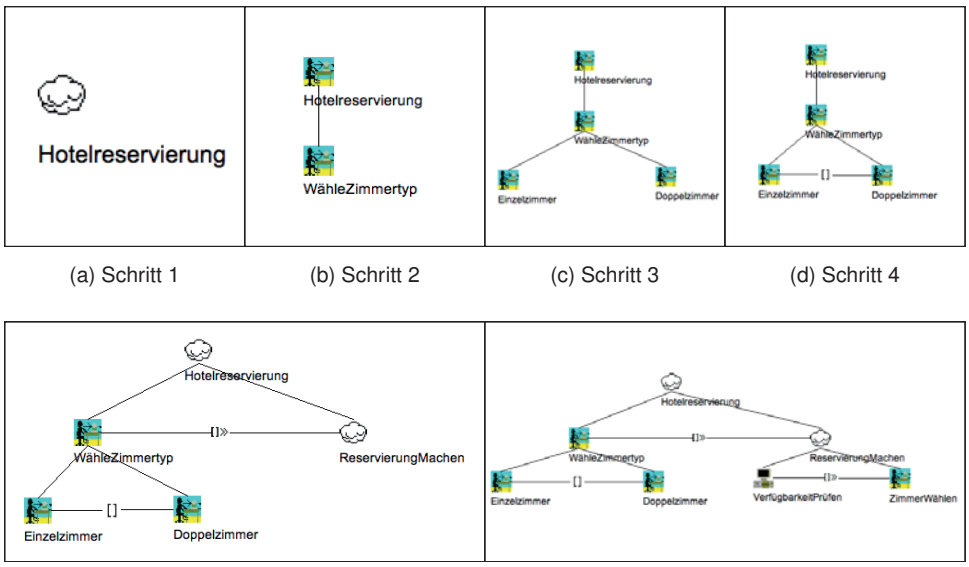


Abbildung 2.2: Blattebene [6]

Zur Veranschaulichung der hierarchischen Strukturierung und Modellierung von Aufgaben, werden Prozess betrachtet, die bei einer Buchung eines Hotelzimmers notwendig sind. Als erste Aufgabe bzw. Task, welches die Wurzel des Baumes bildet, steht die *Hotelreservierung* (s. Abb. 2.3 a).



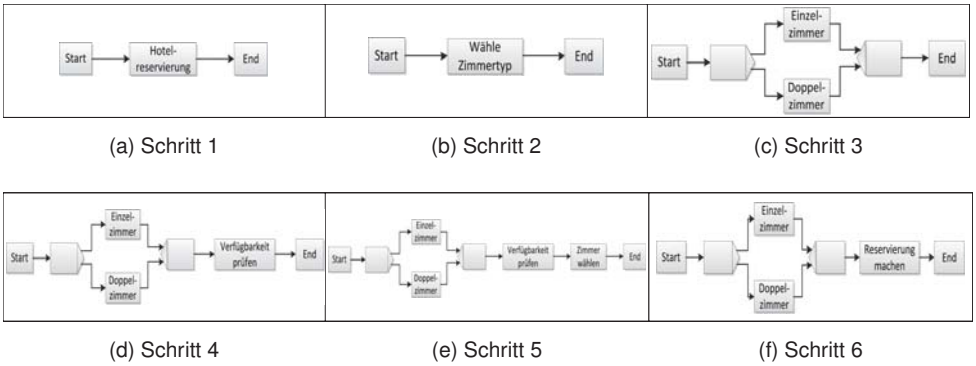
(a) Schritt 1 (b) Schritt 2 (c) Schritt 3 (d) Schritt 4

(e) Schritt 5 (f) Schritt 6

Abbildung 2.3: Ablauf der Modellierung [7]

Die Abbildung 2.3 (b) vermittelt auf den ersten Blick, dass es sich um eine sequentielle Abfolge handeln muss, da sie mit Modellierungen von Sequenzen assoziiert wird, die man vermutlich bereits aus anderen Notationen kennt. Es ist von großer Bedeutung nicht zu vergessen, dass die Hierarchie, die in diesem Bild veranschaulicht wird, keine Sequenz repräsentiert. Der Typ der neuen Aufgabe ist ein Interactiontask, wodurch auch die Aufgabe *Hotelreservierung* nicht mehr abstrakt ist. Diese Veränderung ist auf die technische Modellierungsumgebung CTTE (s. Abschn. 2.4) zurückzuführen, die automatisch die Aufgabentypen während des Zusammenstellens des CTT-Modells anpasst. In unserem Fall wird der Typ der Oberaufgabe an den Typen der Unteraufgabe angepasst.
Im weiteren Verlauf wird die Aufgabe *Wählezimmer* spezifiziert. Dabei werden die Unteraufgaben *Einzelzimmer* und *Doppelzimmer* (s. Abb. 2.3 c) eingeführt. Beide Tasks sind vom Typ Interaktion, da der spätere Benutzer sich entscheidet und seine Auswahl in der zu entwickelnden Anwendung angibt. Bei jeder Verfeinerung einer Aufgabe in Unteraufgaben ist festzustellen, dass das CTT-Modell um jeweils eine weitere Ebene tiefer wird. Um Entscheidungen in CTTs darzustellen, gibt es den bereits oben vorgestellten Deferred Choice-Operator, der zwischen den zwei Zimmertypen angebracht wird (s. Abb. 2.3 d). Der Anwender kann in diesem Fall sich für einen der zwei Zimmertypen entscheiden und auswählen.
Dennoch ist es ein gut geeignetes Beispiel, um zu zeigen, dass mittels CTT Aktivitäten bzw. Aufgaben auf einer sehr intuitiven Art und Weise hierarchisch strukturiert und modelliert werden können.

Um die Verständlichkeit und Simplizität des CTT-Modells abwägen zu können, wird im Vergleich dazu dasselbe Beispiel mit Hilfe der Modellierungssprache ADEPT umgesetzt. Begonnen wird wie im CTT-Baum mit dem Knoten *Hotelreservierung*, wobei in ADEPT zu Beginn eines Prozessmodells ein Startknoten gesetzt wird, der einen korrespondierenden Endknoten besitzt (s. Abb. 2.4 a). Diese Abgeschlossenheit werden wir noch in weiteren ADEPT-Elementen wiederfinden. In der darauffolgenden Abbildung 2.4 b) wird der Knoten *Hotelreservierung* durch den Knoten *WähleZimmertyp* ersetzt. Alternativ gibt es die Möglichkeit *WähleZimmertyp* als Subrpozess darzustellen, die im späteren Verlauf in Betracht genommen wird. Aufgrund dieser Veränderung sollte klargestellt werden, dass die *Hotelreservierung* keine Aktivität im eigentlichen Sinne enthält, d.h. keine Funktion ausdrückt. Auch der Knoten *WähleZimmertyp* wird durch den nächsten Schritt nicht weiter berücksichtigt und "ausgetaucht" (s. Abb. 2.4 c). ADEPT stellt sogenannte XOR-Splits und zugehörende XOR-Joins zur Verfügung, um einen Deferred Choice zu realisieren (s. Abschn. 2.2). Die Knoten *Einzelzimmer* und *Doppelzimmer*, die zur Auswahl stehen sollen, werden zwischen diese XOR-Elemente gesetzt. Wie man vielleicht bereits vermutet, stellt der Knoten *ReservierungMachen* keine wirkliche Aufgabe dar und wird durch die Knoten *VerfügbarkeitPrüfen* und *ZimmerWählen* dargestellt (s. Abb. 2.4 f). Um das Beispiel übersichtlich zu gestalten, wird die Übergabe der Information zwischen den beiden Knoten vernachlässigt.



(a) Schritt 1 (b) Schritt 2 (c) Schritt 3

(d) Schritt 4 (e) Schritt 5 (f) Schritt 6

Abbildung 2.4: Ablauf der Modellierung

Es gibt noch eine andere Möglichkeit in ADEPT das Beispiel abzubilden. In dieser Variante soll auch mit Subprozessen gearbeitet werden, um die Modellierung nah am CTT-Modell zu orientieren. Ober- bzw. Superprozesse, die den Oberaufgaben im CTT-Modell entsprechen, sind mit einem + -Zeichen gekennzeichnet (s. Abb. 2.5). Diese speziellen Knoten enthalten sogenannte Subprozesse, die die Unteraufgaben im CTT-Modell beschreiben und in denen die Spezifizierung der je-

weiligen Superprozesse erfolgt.

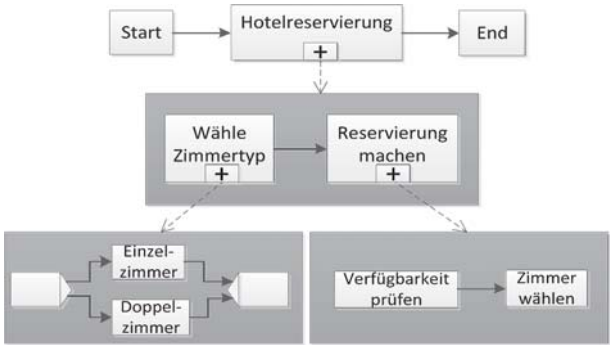


Abbildung 2.5: ADEPT-Modell mit Subprozessen

Zusammenfassend kann festgehalten werden, dass das CTT-Modell intuitiv und verständlich ist, obwohl nicht sofort offensichtlich ist, dass Oberaufgaben keine wirkliche Funktion haben, folglich nicht ausgeführt werden, sondern auf der darunterliegenden Ebenen ausführlicher spezifiziert werden. Die eigentlichen ausgeführten Aktivitäten sind in den Blättern des Baumes beschrieben. Ein weiterer Vorteil ist die implizite Darstellung der Informationsübergabe, die durch spezielle Operatoren ausgedrückt werden. Das reduziert zusätzlichen Modellieraufwand und garantiert bessere Überschaubarkeit bei der Modellierung.
Im ersten ADEPT-Modell sind statt den Oberaufgaben nur die Blätter dargestellt. Es werden also nur die ausführbaren Aktivitäten modelliert, wodurch der Fokus auf die relevanten Aufgaben gelenkt ist. Es ist daher kein Rückschluss auf den ursprünglichen CTT-Baum möglich, von dem aus das ADEPT-Modell erstellt wurde. Des Weiteren gibt es bei ADEPT keine speziellen Aufgabentypen, denen die Aufgaben zugeordnet werden können.
Durch die Verwendung von Subprozessen im zweiten ADEPT-Modell wird die Ähnlichkeit zur baumartigen Struktur des CTT erreicht. Der Entwurf des entsprechenden CTT-Baumes aus dieser Darstellung ist problemlos rekonstruierbar. Im Gegensatz zum CTT-Modell kann die Baumstruktur zum Nachteil werden, da die bei größeren und komplexeren Modellen schnell die Beziehungen zwischen den Super- und Subprozessen verloren gehen und dabei die wesentlichen Aufgaben nicht identifiziert werden können.

2.4 Concurrent Task Trees Environment

Das *ConcurTaskTrees Environment* (CTTE) [8] ist der graphische Editor zum Modellieren und zur Analyse von CTT-Modellen. Dieses Open-Source-Programm

kann über [9] bezogen werden. Der wichtigste Bestandteil des Programms ist der Aufgabeneditor, mit dessen Hilfe verschiedene Aufgabenmodelle, wie das obige Beispiel der Hotelreservierung (s. Abb. 2.2), erzeugt werden können. Des Weiteren sind eine Reihe von Funktionalitäten vorhanden, die nun im Folgenden beschrieben werden sollen.

Main View

In der Hauptansicht (Main View) des CTTE kann der gewünschte Baum anhand der verfügbaren Aufgabentypen und temporalen Operatoren modelliert werden (s. Abb. 2.6). Die Eigenschaften einer Aufgabe können per Rechtsklick auf die jeweilige Aufgabe textuell oder durch Auswahlboxen bearbeitet werden. Eine Aufgabe wird im Wesentlichen durch einen *Identifier* und *Namen* beschrieben. Der Identifier ist frei wählbar und sollte eindeutig sein. Der Name einer Aufgabe dient, im Gegensatz zum Identifier, zur Beschreibung, um diesen von Anderen unterscheiden zu können. Mit der Angabe der *Frequenz* wird ermöglicht anzugeben, wie oft eine Aufgabe ausgeführt wird. Zur Auswahl stehen die verschiedenen Werte *niedrig*, *mittel* und *hoch*. Neben der Frequenz gibt es die Möglichkeit, *Objekte* einer Aufgabe zu definieren. Aufgrund eines fehlenden Objektmodells, spielt es keine Rolle, dass bspw. Aufgabe A ein Objekt O vom Typ T benötigt; die Angaben werden auch nicht mit anderen Aufgaben und deren Anforderungen an Objekte abgeglichen. Wenn in Aufgabe B definiert wird, dass auf ein in Aufgabe X erzeugtes oder modifiziertes Objekt zugegriffen wird, ist es nicht von Bedeutung, ob Aufgabe C tatsächlich auf diesem Objekt operiert. Ähnlich wie bei den benötigten Objekten erfolgt die Angabe einer (oder auch mehrerer) *Vorbedingung(en)* ähnlich den Objekten rein textuell; auch in diesem Fall gibt es keinerlei semantischen Zusammenhang zwischen Vorbedingung, Aufgabe und dem Modell insgesamt. Die Prüfung, ob die Vorbedingung(en) für eine Aufgabe bei Eintritt gegeben sind, obliegt dem Entwickler des Modells.
Mit Hilfe der Main View lassen sich nicht nur Arbeitsabläufe eines Benutzers innerhalb eines Systems beschreiben, sondern auch Abläufe, an denen mehrere Benutzer beteiligt sind. Hierzu wird der *Cooperative Mode* bereitgestellt, in dem der *Cooperationtask* zur Verfügung gestellt wird. In diesem Modus können Aufgaben mit Rollen versehen werden, um festzulegen, wer sie ausführen darf. Das Vorgehen zur Modellierung geschieht dabei top-down: beginnend bei der Wurzel des kooperativen Modus werden die Aufgaben immer weiter verfeinert, bis sich die Aufgaben nur noch auf einzelne Rollen beziehen. Die Vorteile dieses Ansatzes beruhen vor allem auf der graphischen Syntax der CTT-Modelle: die Beschreibung

von Rollen und Kooperation in jeweils eigenen CTT-Modellen hält das Gesamtmodell vergleichsweise übersichtlich und damit wartbar. Zudem ist es ohne weiteren Aufwand möglich, jede Rolle mit all ihren Aufgaben einzeln zu betrachten und bei Bedarf auch zu bearbeiten. Auch kann über entsprechende Relationen im kooperativen CTT-Modell der Austausch von Informationen dargestellt werden. Gleichzeitig ergibt sich die wesentliche Schwäche des Ansatzes aus genau jener separaten Beschreibung der Interaktion. Direkte Interaktion, d.h. dass eine Aufgabe einer Rolle unmittelbar aus einer anderen Rolle heraus gestartet wird, ist mittels des kooperativen CTT-Modells nicht darstellbar. Ein weiterer Schwachpunkt kooperativer CTT-Modelle ist die Aufsplittung des Modells in die einzelnen Rollen- und das Kooperationsmodell selbst. Ähnlich wie bei UML ergibt sich hier das Gesamtmodell erst durch das Zusammensetzen der einzelnen CTT-Modelle.

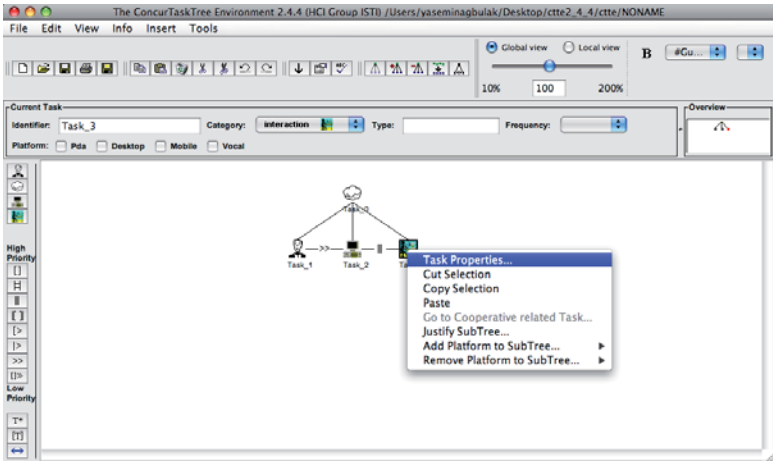


Abbildung 2.6: Main View

Informal To Formal Description

Mit Hilfe dieses Werkzeugs wird es dem Benutzer ermöglicht, Aufgabenmodelle aus Anwendungsfällen bzw. Szenariobeschreibungen, die in Textform beschrieben werden, zu generieren (s. Abb. 2.7). Allerdings erfolgt dies nicht automatisch, da der Benutzer Begriffe oder Textpassagen aus den textuellen Beschreibungen wählt, um diese dann mit Hilfe von entsprechender Buttons (Add, Delete) wahlweise als *Rolle*, *Aufgabe* oder *Objekt* zu deklarieren. Bei der Beschreibung von Aufgaben, muss einer Aufgabe eine Rolle zugeordnet werden und ein entsprechender Aufgabentyp ausgewählt werden. Auf diese Weise können Aufgabenmodelle ohne temporale Operatoren erzeugt werden.

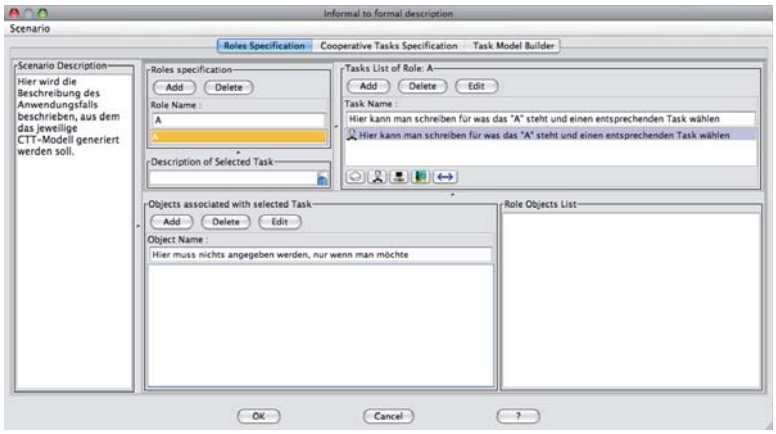


Abbildung 2.7: Funktionalität: Informal To Formal Description

Erreichbarkeitsanalyse

Unter Verwendung der Erreichbarkeitsanalyse kann automatisiert geprüft werden, ob eine bestimmte Aufgabe ausgehend von einer anderen festzulegenden Aufgabe anhand der im Modell gewählten temporalen Operatoren erreichbar ist. Etwas formaler ausgedrückt wird überprüft, ob es einen Pfad von einem Knoten des Baumes zu einem anderen Knoten gibt. Es kann auch eine zusätzliche Aufgabe angegeben werden, die auf diesem Pfad liegen muss.

Simulator

Die Anwendung des integrierten Simulators erlaubt die möglichen Verläufe des Aufgabenmodells semi-automatisch zu überprüfen (s. Abb. 2.8). Der CTT-Baum wird bei der Überprüfung auf syntaktische Fehler, wie fehlende Operatoren oder Teilaufgaben, untersucht. Mit dem Simulator kann, wie der Name schon sagt, die Ausführung eines Aufgabenmodells simuliert werden. Diese Funktion ist sowohl für normale als auch für kooperative Modelle verfügbar. Mit einem Klick auf eine Aufgabe innerhalb des Strukturbaumes erscheint ein Cursor auf dieser und mit einem weiteren Klick auf dieselbe Aufgabe springt der Cursor auf die nächste Aufgabe in der Ausführungsreihenfolge. Damit kann die Abarbeitung der Aufgaben im Baum kontrolliert werden. In der Simulation ist es aber bisher nicht möglich, Daten des Systems anzuzeigen oder Eingaben zu machen. Diese Aufgaben werden also, wie bereits oben erwähnt, ohne Berücksichtigung weiterer Angaben wie Vorbedingungen und/oder benötigten Objekten ausgewählt und ausgeführt. Mit Hilfe der Simulation des CTT-Modells wird es ermöglicht, dem Endbenutzer die Abläufe

der Aufgaben zu verdeutlichen und ein besseres Verständnis für das CTT-Modell zu bekommen.

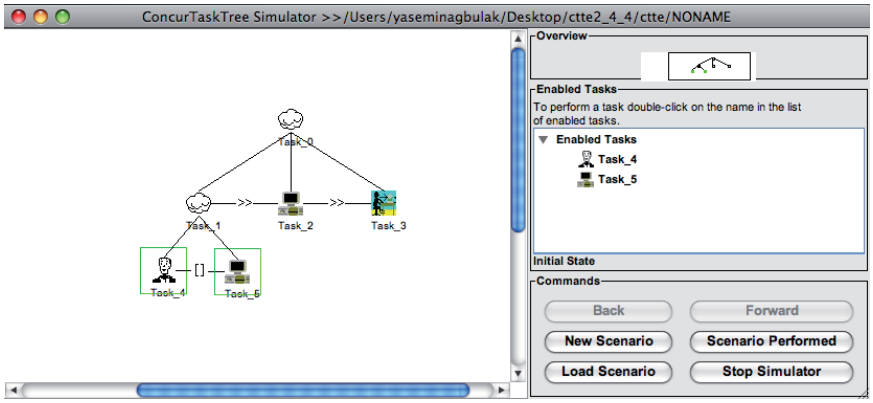


Abbildung 2.8: Funktionalität: Model Simulator

Bewertung

Mit Hilfe des CTTE-Simulators wird das CTT-Modell auf Fehler überprüft und unterstützt somit den Endbenutzer bei der syntaktisch korrekten Modellierung von CTT-Modellen. Bei der Durchführung der Simulation ist hervorzuheben, dass durch einen Cursor das Verständnis verbessert wird. Dabei springt der Cursor auf die jeweils auszuführende Aufgabe und verdeutlicht dadurch die Ausführungsreihenfolge der Aufgaben.
Leider treten in CTTE immer wieder “überraschende” Effekte auf z.B. wenn man nachträglich Aufgaben hinzufügt und diese an die erste Stelle verschiebt, kann man keine zeitliche Relation zur folgenden Aufgabe festlegen, da sie scheinbar noch immer als letzte bzw. als erste Aufgabe angesehen wird. Diese Option kann in der Main View vom Modellierer gewählt werden. Um die gewünschte Position des neuen Knotens zu erhalten, müssen die Identifier der jeweiligen Aufgaben manuell geändert bzw. umbenannt werden, da dies das CTTE nicht unterstützt. Bei einer Sequenz von drei Knoten ist es noch machbar, die jeweiligen Knoten entsprechend umzubenennen. Je mehr Knoten hinter dem eingefügten Knoten folgen, desto komplexer und aufwändiger wird die Änderung der betroffenen Knoten. Ein weiterer negativer Aspekt ist, dass die Verbindungsrichtung der temporalen Operatoren nur unidirektional ist. Außerdem werden weder Aussagen über irgendwelche Informationskontrollen noch Rückmeldungen in Aufgaben und in der Kommunikation von verschiedenen Rollen gegeben. Das Konzept der Beschreibung von Vorbedingungen, die für eine Aufgabe optional angegeben werden kann, ist

nur textuell möglich, welches sich in einer graphischen Modellierungsumgebung als nachteilhaft erweist.

3 Ändern von Prozessmodellen durch CTT

Verschiedene Aufgabenstellungen und Situationen, die sich mit der Zeit zunehmend verändern, verändern dadurch auch die zunehmend abwechslungsreicher werdenden Wünsche und Anforderungen der Endbenutzer hinsichtlich Anwendungssystemen. Somit sind die zugrundeliegenden Prozessmodelle, die die Anwendungslogik widerspiegeln, immer wieder Veränderungen unterworfen und müssen an neue Gegebenheiten angepasst werden. Die Durchführung solcher Änderungen muss so einfach wie möglich gehalten werden und darf niemals zu Konsistenz- oder Korrektheitsverletzungen führen. Das bedeutet, dass prozessorientierte Anwendungen auch im Anschluss an Änderungen robust und stabil laufen müssen [10]. Aspekte wie Flexibilität und Adaptivität sind in diesem Zusammenhang von großer Bedeutung.
In diesem Kapitel betrachten wir einige sogenannte *Process Change Pattern* [10] angewandt auf CTT- und ADEPT-Modelle. Change Pattern sind bewährte Lösungs-Schablonen für wiederkehrende Entwurfsprobleme im Bereich von Prozessmodellierungssprachen, ähnlich wie Design Pattern, die in der Softwarearchitektur und Softwareentwicklung zum Einsatz kommen [11]. Sie stellen damit eine wiederverwendbare Vorlage zur Problemlösung dar, die in einem bestimmten Zusammenhang einsetzbar ist. Dadurch reduzieren sie die Komplexität von Prozessänderungen (wie z.B. Design Pattern in der Systementwicklung eingesetzt werden, um die Komplexität des Systems zu reduzieren [11]) und ermöglichen Änderungen auf einer hohen Abstraktionsebene in Form von einzelnen Knoten- und Kantenoperationen auszudrücken [12]. In den nachfolgenden Abschnitten werden die Change Pattern *Insert*, *Delete*, *Replace* sowie *Move* näher betrachtet.

3.1 Anwendung des Change Pattern Insert

Das Change Pattern *Insert* kann verwendet werden, um ein Prozessfragment bzw. ein Knoten in ein bereits bestehendes Prozessmodell einzufügen. Das Einfügen eines Knotens X zwischen die Knoten A und C im ADEPT-Modell stellt die Abbildung 3.1 a) dar. Für die Umsetzung dieser Ad-hoc-Änderung wird die Kante zwischen A und C gelöscht und der Knoten X eingefügt. Um die neue Sequenz zu

vervollständigen, werden die einzelnen Knoten durch entsprechende Kanten wieder verbunden. Das Resultat ist eine neu entstandene Sequenz AXC (s. Abb. 3.1 b). In diesem Fall spricht man von einem *Serial Insert*.
Die Umsetzung des Insert Pattern im CTT-Modell illustriert die Abbildung 3.1 d). Das Einfügen des neuen Knotens X erfordert dieselben Änderungen wie im ADEPT-Modell. Soll X zwischen die zwei Geschwisterknoten A und C eingefügt werden, so wird zunächst die bestehende Kante zwischen diesen aufgehoben. Der Knoten X wird zwischen die Blätter A und C gesetzt und natürlich muss auch hier die fehlenden Kanten, sowohl zwischen A und X, als auch zwischen X und C gezogen werden.
In beiden Modellen ist die Realisierung des Serial Inserts problemlos und intuitiv umsetzbar.

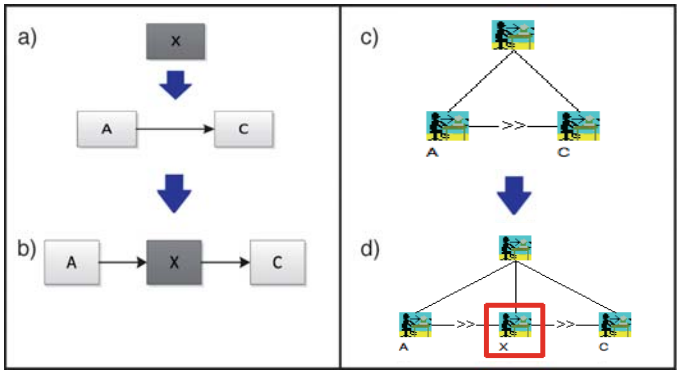


Abbildung 3.1: Einfügen in die Sequenz

Eine weitere Funktion des Insert Pattern ist die Erzeugung einer Parallelität durch das Einfügen eines neuen Knotens (*Parallel Insert*). Betrachten wir nun die Sequenz ABC in Abbildung 3.2 a). Als erstes werden alle Kanten entfernt und der Knoten X eingefügt. Damit die Knoten B und X parallel ablaufen können, wird ein AND-Split vor dem Knoten B und ein korrespondierender AND-Join nach dem Knoten X benötigt (s. Abb. 3.2 b). Anschließend müssen noch die fehlenden Kanten zwischen den jeweiligen Knoten angebracht werden. Dabei ist zu beachten, dass aus dem AND-Split zwei Kanten ausgehen, jeweils eine für X und für B, und zwei Kanten in den AND-Join eingehen, die von X und B ausgehen.
Um nun in CTT die Knoten B und X nebenläufig ausführen zu können, werden zuerst die temporalen Relationen zwischen A, B und C entfernt. Danach wird ein leerer Knoten zwischen A und C eingefügt. Dabei wird B auf eine neue Blattebene verschoben, wo auch der neue Knoten X hinzugefügt wird. B und X sind mit dem leeren Knoten verbunden. Mit dem Hinzukommen dieser beiden Kindknoten, wird die Ebene (s. Abb 3.2 d) des CTT-Modells um eine neue erweitert. Als letz-

tes werden noch die entsprechenden temporalen Relationen zwischen den Knoten wieder angefügt. Den zentralen Ausgangspunkt für die eigentliche Parallelität bildet der entsprechende Concurrent-Operator zwischen den Geschwisterknoten B und X.
Die Bewertung des CTT-Modells in Abbildung 3.2 d) hinsichtlich der Verständlichkeit und Überschaubarkeit - durch die Zerlegung und Verfeinerung von Knoten auf mehrere Ebenen - liegt sehr hoch. Jedoch ist zu beachten, dass bei größeren und komplexeren Modellen mit mehreren Ebenen es schnell passieren kann, dass der Überblick über die Zusammenhänge bzw. die temporalen Beziehungen zwischen den einzelnen Ebenen und Knoten verloren geht.

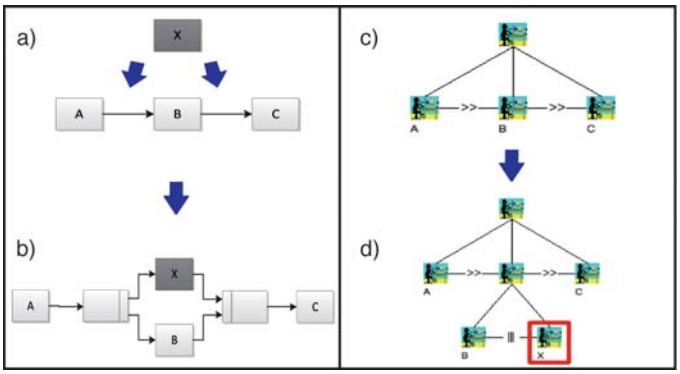


Abbildung 3.2: Paralleles Einfügen eines Knoten

Die Anwendung des Insert Pattern beschränkt sich nicht auf sequentielle und parallele Abschnitte eines Prozesses, sondern kann auf beliebigen Prozessbereichen und Ebenen Verwendung finden. Eine weitere Art des Insert Pattern ist der *Conditional Insert*. Der neu hinzugekommene Prozessfragment ist mit einer Ausführungsbedingung verbunden. Das Einfügen eines neuen Knotens in den verzweigten Teil des Prozessmodells veranschaulicht die Abbildung 3.3. Der Ablauf des gegebenen Prozesses beginnt mit der Ausführung des Knotens A, dem ein XOR-Split folgt. Angekommen an dieser Stelle muss zwischen B und C entschieden werden. Der Verzweigungspfad endet nach dem Ausführen des jeweils gewählten Knotens mit dem XOR-Join und der komplette Prozessablauf resultiert mit dem Knoten D (s. Abb. 3.3 a).
Für das Hinzukommen von X im ADEPT-Modell, wird die Kante zwischen B und dem XOR-Join gelöscht, der neue Knoten zwischen diese eingefügt und als letztes die fehlenden Kanten sowie zwischen B und X, als auch zwischen X und dem XOR-Join angelegt (s. Abb. 3.3 b). Die Realisierung des Insert Patterns ist aus technischer Sicht ohne großen Aufwand und analog zum Einfügen einer Sequenz. Die Änderung im CTT-Modell zieht eine größere Änderung der Baumstruktur nach

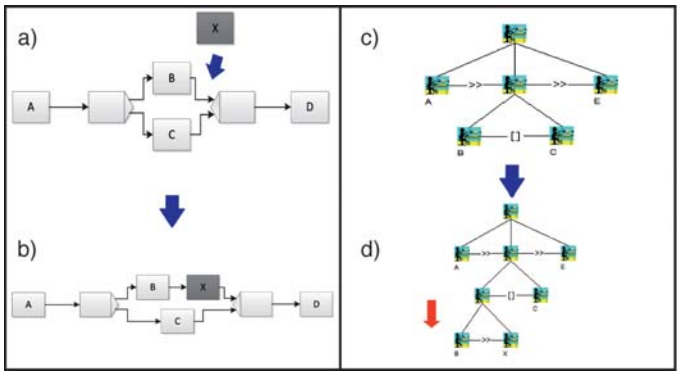


Abbildung 3.3: Einfügen eines Elements innerhalb einer Verzweigung

sich. Damit der neue Knoten in das CTT-Modell eingebettet werden kann, werden wieder zunächst die bestehenden temporalen Relationen zwischen den Knoten entfernt. An die Stelle von B wird ein leerer Knoten eingefügt und der Knoten B auf eine neue Blattebene verschoben. Dem leeren Vaterknoten wird nun B und der neue Knoten X zugeordnet. Nachdem der neue Knoten angefügt ist, werden die vorherigen temporalen Relationen wieder eingeügt. Durch das Anbringen des Enabling-Operators zwischen B und X wird der sequentielle Ablauf dieser beiden Blätter realisiert.

Ohne die Verwendung eines Vaterknotens und einer neuen Blattebene, müsste man die Knoten B, X und C auf ein und derselben Blattebene darstellen mit den entsprechenden Operatoren, d.h. zwischen B und X ein Enabling-Operator und zwischen X und C ein Choice-Operator. Syntaktisch gesehen wäre diese Weise der Modellierung korrekt, aber die eigentliche Semantik, die mit dem Einfügen von X erreicht werden sollte, würde sich verändern. Die Ausführungsreihenfolge der Knoten im Baum würde dann so aussehen: Nach dem Knoten A würde B folgen und dann hätte man die Möglichkeit zwischen X und C zu entscheiden. Durch die Erweiterung des CTT-Modells um eine weitere Blattebene führt dazu, dass zwischen der Sequenz BX und dem Knoten C eine Auswahl getroffen werden kann, wodurch die geforderte Semantik erreicht wird.

Im Vergleich zum ADEPT-Modell lässt sich schließen, dass das CTT-Modell in diesem Fall aufgrund der Verständlichkeit und der Übersichtlichkeit definitiv schlechter abschneidet als das ADEPT-Modell. Während im ADEPT-Modell auf einer Ebene bzw. innerhalb ein und derselben Verzweigung die Änderung vorgenommen werden kann, wird im CTT-Modell eine weitere Ebene benötigt und damit die Baumstruktur noch tiefer. Je mehr Ebenen im CTT-Modell hinzukommen, desto schwieriger wird es, die semantischen Beziehungen zwischen den verschiedenen Ebenen im Auge zu behalten und zu verstehen.

3.2 Anwendung des Change Pattern Delete

Das Löschen von einem Element aus einer Sequenz kann sowohl in ADEPT als auch in CTT sehr einfach umgesetzt werden. Dazu gibt es das sogenannte *Delete Pattern*. Durch das Löschen des Knotens B ändert sich in beiden Modellierungen die ursprüngliche Struktur des gegebenen Modells bzw. Baumes nicht, d.h. es kommen keine weiteren Ebenen hinzu oder fallen weg (s. Abb. 3.4 c). Sobald in CTT der Knoten B gelöscht wird, wird die temporale Relation, die nach B folgt, mitgelöscht. D.h. es wird der Operator von A übernommen und verbindet somit A mit C (s. Abb. 3.4 d).

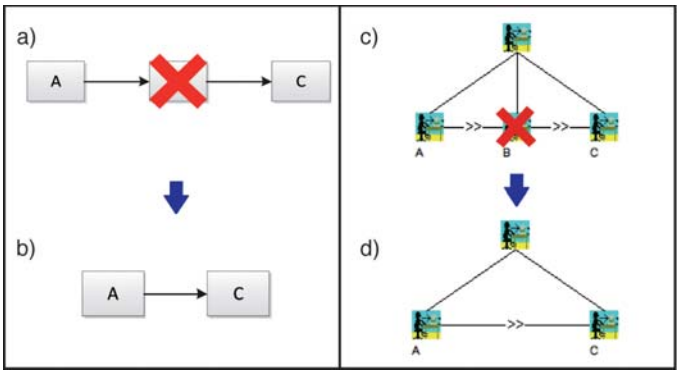


Abbildung 3.4: Löschen aus der Sequenz

Eine weitere Operation ist das Löschen von Knoten aus Verzweigungen. Abbildung 3.5 a) zeigt, wie in ADEPT ein Knoten aus einer AND-Block gelöscht wird. Nachdem der Knoten B gelöscht wird, der vorher mit dem Knoten C parallel ausgeführt werden konnte, entsteht nun eine Sequenz ACD. Da durch das Entfernen von B die Parallelität aufgehoben wird, wird der umschließende AND-Block aus dem Modell ebenso entfernt (s. Abb. 3.5 b). Stellt man sich nun das Modell nicht mit AND-Elementen , sondern mit XOR-Elementen vor, so verbleibt auf dem Pfad von B ein leerer Zweig. Je nachdem welchen Pfad man wählen würde, würde entweder die Ausführungsreihenfolge AD oder ACD resultieren.
Dieselbe Operation wird im CTT-Modell (s. Abb. 3.5 c) durchgeführt. Die Ebene, auf der sich die parallele Ausführung von B und C befindet, wird durch das Löschen von B aufgehoben. Dabei wird der Knoten C auf die obere Ebene zwischen A und D eingefügt, da in CTT streng definiert ist, dass ein Element mindestens zwei Kindknoten besitzen muss. Nachdem C eine Ebene hochrückt, entsteht die Sequenz ACD als Ergebnis (s. Abb. 3.5 d). Würde man auch hier nicht von einer parallelen Ausführung von B und C ausgehen, sondern von einer alternativen Ausführung, so sehe das Resultat, wie auch in ADEPT, anders aus. Statt dem Knoten

B wird ein leeres Blatt angebracht. So bleibt die unterste Ebene mit dem leeren Blatt und mit C erhalten und je nach Entscheidung entsteht dann die Abfolge AD oder ACD.

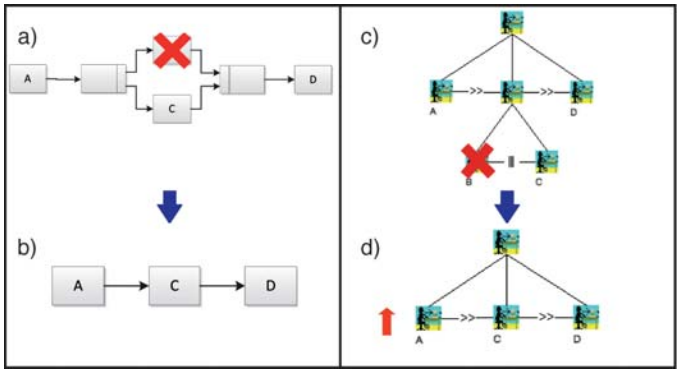


Abbildung 3.5: Löschen innerhalb einer Verzweigung

Sollte man diese beiden Modellierungen von ADEPT und CTT vergleichen, kann man sagen, dass in beiden Fällen die Modelle nach der Ausführung der Lösch-Operation vereinfachter und überschaubarer werden. Allerdings muss beachtet werden, dass hier nur sehr kleine und einfache Beispiele in Betracht gezogen werden. Mit steigender Komplexität steigt auch der Schwierigkeitsgrad bei der Modellierung des neuen bzw. veränderten Zustands.

3.3 Weitere Change Pattern

In diesem Unterabschnitt werden weitere Change Pattern vorgestellt und die Umsetzung mit den jeweiligen Modellierungssprachen untersucht und diskutiert.
Das *Replace Pattern* erlaubt das Ersetzen eines Prozessfragments durch ein anderes. Die Realisierung dieses Patterns kann durch die bereits vorgestellten Pattern Löschen und Einfügen erreicht werden. In CTT kann das zu ersetzende Element zwar gelöscht werden, aber das neue Element wird immer nur ganz rechts eingefügt. Dies verändert folglich die gewünschte Reihenfolge der einzelnen Prozesselemente. Um dieses Problem zu umgehen, sollte man das zu ersetzende Element umgeändert werden, d.h. der Name und die Eigenschaften des neuen Elements können so übernommen werden, ohne löschen und einfügen (s. Abb. 3.6 d).
Ein weiterer Adaptation Pattern ist das Parallelisieren eines ausgewählten Prozessfragments. Die folgende Abbildung 3.7 zeigt, wie die Parallelisierung der ausgewählten Knoten B, C und D in ADEPT und in CTT aussehen.
Damit die gewünschten Elemente parallel ablaufen können, wird in ADEPT der

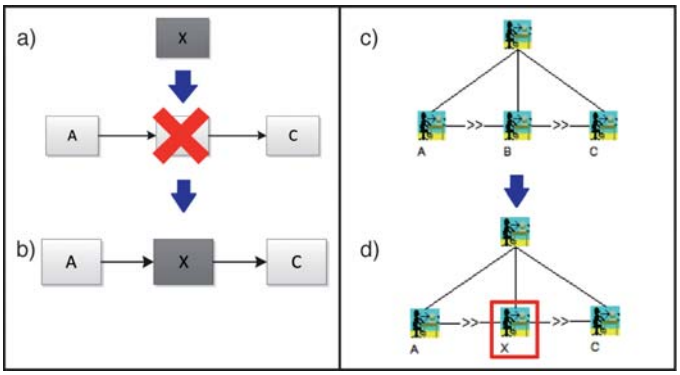


Abbildung 3.6: Altes Element wird durch neues ersetzt

AND-Knoten benötigt. Zu Beginn und am Ende der zu parallelisierenden Elemente wird dieser spezielle Knoten eingefügt und somit das gewünschte Ergebnis erzielt (s. Abb. 3.7 b).

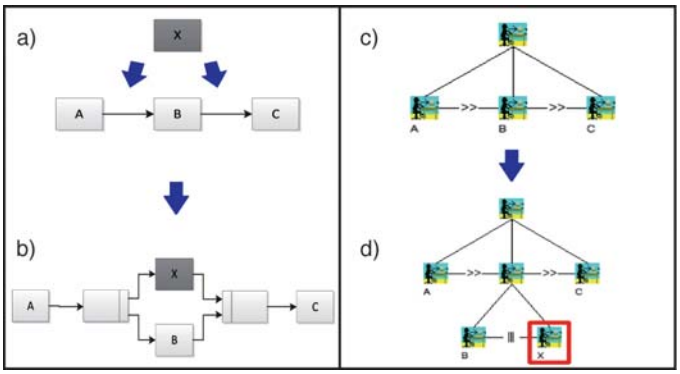


Abbildung 3.7: Auswahl wird parallelisiert

Während beim ADEPT-Modell immer auf einer Ebene gearbeitet wird, sieht es beim CTT anders aus. Es wird eine neue Blattebene angelegt, die die zu parallelisierenden Knoten B, C und D beinhaltet. Der spezielle Concurrent-Operator zwischen diesen Elementen führt dazu, dass alle drei gleichzeitig ablaufen können.

Ein letzter Change Pattern, den wir noch einführen möchten, ist der sogenannten *Move Process Fragment Pattern*. Mit Hilfe des *Move Process Fragment Pattern* kann ein Prozessfragment seiner aktuellen Position in eine neue verschoben werden. Alternativ kann das Move Pattern durch die Kombination vom *Insert* und *Delete* Pattern oder auf Basis von primitiven Änderungen realisiert werden, doch in unserem Fall führen wir es als eigenständigen Pattern ein, da es den Modellierern bzw. Benutzern eine höhere Abstraktionsebene bietet. Im Folgenden betrachten wir drei Umsetzungsmöglichkeiten des vorgestellten Move Patterns.

Die erste Variante wird in der Abbildung 3.8 betrachtet, bei der das zu verschiebende Prozessfragment bzw. Knoten A zwischen zwei direkt aufeinanderfolgende Knoten wieder eingebettet wird. Formell spricht man in diesem Fall von einer *seriellen* Verschiebung bzw. vom *serial move*, da A in eine Sequenz platziert wird.
Die Umsetzung dieses Patterns mit Hilfe von ADEPT wird in Abbildung 3.8 b) illustriert. Das zu verschiebende Element A wird innerhalb der XOR-Komponenten eingefügt. Die Realisierung ist mit dem Löschen des Knotens A aus seiner ursprünglichen Position, sowie mit dem erneuten Einfügen an seine neue Stelle verbunden.
Während in ADEPT die Umstellung des Knotens A mit geringem Aufwand erzielt wird, so ist sie im CTT doch wesentlich komplizierter. Das Verschieben von A bewirkt, dass das CTT-Modell in diesem Fall um eine Blattebene erweitert wird (s. Abb. 3.8 d). Um die gewünschte Semantik des Prozesses beizubehalten, ist es unvermeidlich, die Knoten A und D auf eine neue Blattebene zu verlegen und den entsprechenden Operator einzufügen. Den Knoten A auf die gleiche Blattebene wie C und D mit den jeweiligen Operatoren zu setzen, würde bedeuten, dass nachdem A ausgeführt wird eine Auswahl zwischen C und D getroffen werden kann. Diese Tatsache würde die gewünschte semantische Bedeutung des Prozesses verfälschen.

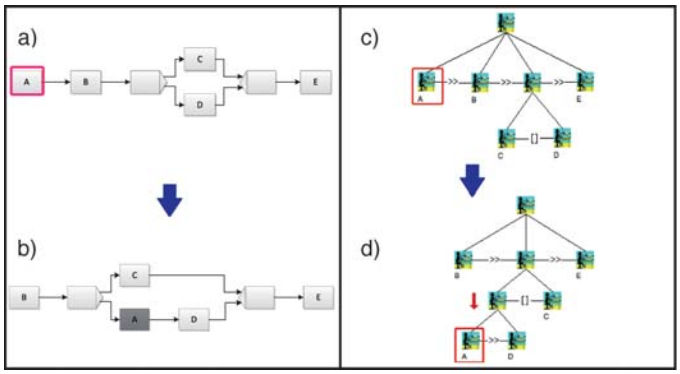


Abbildung 3.8: Verschieben eines Elements

Eine andere Möglichkeit des Move Pattern ist das Verschieben des Knotens innerhalb einer Auswahl (XOR-Verzweigung bzw. Choice), daher auch der Begriff *Conditional Move*. In ADEPT sieht die Gestaltung des Patterns sehr simpel aus: Eine neue ausgehende Kante von XOR-Split, der verschobene Knoten A und eine neue eingehende Kante zum XOR-Join wird hinzugefügt (s. Abb. 3.9 b). Auch die Realisierung mittels CTT ist ohne besonderen Aufwand umsetzbar. Der Knoten A wird auf der Blattebene von C und D eingefügt und durch den Choice-Operator mit Knoten C verbunden (s. Abb. 3.9 d). Da es sich hierbei um eine Entscheidung

handelt, spielt es in CTT sowie auch in ADEPT keine Rolle, ob A neben C oder D, oder zwischen diese beiden Knoten gesetzt wird.

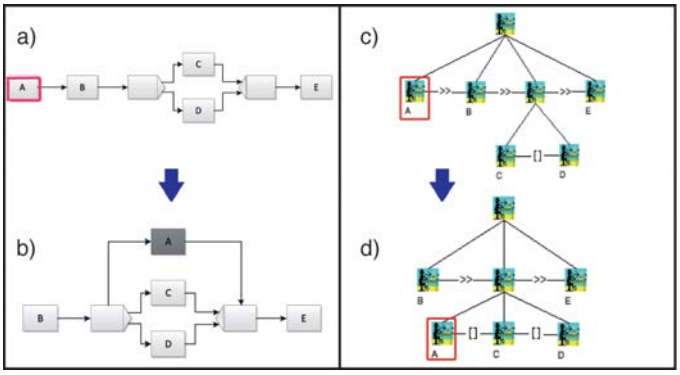


Abbildung 3.9: Verschieben eines Elements

Die dritte und letzte Ausprägung des Move Pattern ist der *Parallel Move*. Der zu verschiebende Knoten wird in diesem Ansatz mit einem anderen Knoten innerhalb einer XOR-Verzweigung parallelisiert. Das heißt, es müssen zwei Operationen vorgenommen werden: Zum einen das Verschieben des Knotens und zum anderen das Hinzufügen von entsprechenden Notationselementen, um eine Parallelität zu gewähleisten.
Abbildung 3.10 b) veranschaulicht das *Parallel Move* Pattern durch die Verwendung von ADEPT. Die Knoten A und C werden synchronisiert, indem Knoten A, sowie die ein- und ausgehenden Kanten von C gelöscht werden. Um den Knoten C werden jeweils die AND-Knoten angebracht und durch Kanten verbunden. In diesen AND-Block kommt der vorhin gelöschte Knoten A hinzu und wird ebenfalls mit dem AND-Split und dem AND-Join durch Kanten verbunden. Das gewünschte Resultat ist jedoch noch nicht erfüllt. Abschließend muss der AND-Block durch Kanten mit dem umschließenden XOR-Block durch Kanten verknüpft werden.
Das entsprechende CTT-Modell ist in der Abbildung 3.10 d) zu sehen. Die Wirkung dieses Patterns ist die gleiche wie die des bereits vorgestellten *Serial Move* Patterns. Das CTT-Modell wird durch die Verschiebung um eine neue Ebene tiefer aus demselben Grund wie beim Serial Move Pattern. Der einzige Unterschied ist die temporale Beziehung, die zwischen A und C liegt. Um die beiden Blätter nebenläufig darzustellen, wird der Concurrent-Operator gewählt und eingesetzt.

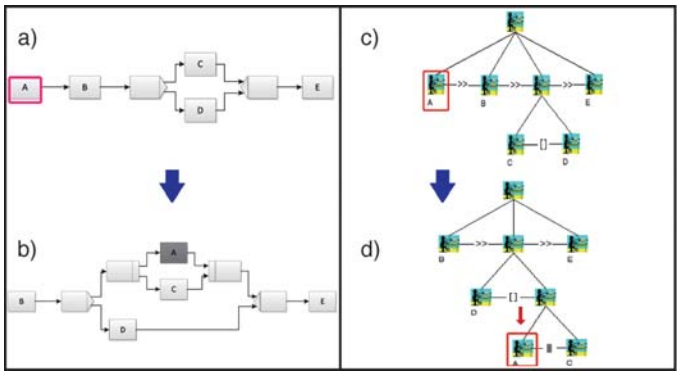


Abbildung 3.10: Verschieben eines Elements

3.4 Zusammenfassung

Die Wirkung der vorgeführten Change Pattern angewandt auf Adept und CTT ist je nach Pattern unterschiedlich.
Sowohl mithilfe von ADEPT als auch von CTT sind die Change Pattern umsetzbar, jedoch mit unterschiedlichen Stufen des Aufwandes. ADEPT begünstigt die Realisierung der Pattern durch ihre “flache” Struktur, ohne dabei viele Grundoperationen (Löschen, Einfügen) auf die einzelnen Elemente (Knoten, Kanten) anwenden zu müssen. Modifikationen verändern Struktur des ursprünglichen Modells nicht beschwerlich, eher nachvollziehbar und leicht verständlich.
Die Change Pattern können in CTT nur eingeschränkt ohne weitere Umständlichkeiten verwendet werden. Eine weitere Problematik ist, dass sich derartige Änderungen auf den gesamten CTT-Modell auswirken und diesen in gewisser Weise umstrukturieren. Dabei kann das CTT-Modell um weitere Ebenen wachsen bzw. schrumpfen. Ein positiver Faktor ist die hierarchische Dekomposition von Aufgaben, welche zu einer leichten Verständlichkeit von Aufgabenmodellen führt. Bei großen Aufgabenmodellen kann dieser Vorteil zu einem Nachteil ausarten, da die temporalen Beziehungen der Aufgaben auf den verschiedenen Ebenen nicht entfallen dürfen.

4 Fallstudie

In diesem Kapitel werden die bisher kennengelernten Elemente und Eigenschaften des CTT anhand einer Fallstudie illustriert. Dazu wird ein Prozess für die Organisation und Ablauf eines *Schachtags* [13] herangezogen, aus dem jeweils unterschiedliche Ausprägungen von CTT-Bäumen resultieren. Gleichzeitig wird der Prozess auf Basis von ADEPT modelliert und mit CTT-Modellen verglichen.
Das Prozessmodell zum Schachtag beginnt mit der Beantragung einer Räumlichkeit, in der das Schachturnier stattfinden soll (s. Abb. 4.1). Nachdem eine Räumlichkeit zur Verfügung steht wird ein Termin für den Schachtag festgelegt und anschließend beginnen die Vorbereitungen für diesen Tag. Der Knoten *Vorbereitungen bis zum Schachtag* stellt in ADEPT eine Subprozess-Aktivität dar. Um nicht von der Betrachtung des eigentlichen Prozesses abzukommen, wird die nähere Untersuchung des zugehörigen Subprozesses auf später verschoben. Sind die Vorbereitungen abgeschlossen findet ein Mitarbeitertreffen statt, welches den organisatorischen Ablauf des Schachtags umfasst. Ist das Organisatorische abgeklärt, beginnt der Aufbau der Essensstationen, wo anschließend Speisen und Getränke verkauft werden. Gleichzeitig zu diesen Aktivitäten werden die für das Turnier benötigten Spielbretter aufgebaut, dann Computer installiert, damit im nächsten Schritt die Anmeldungen der Teilnehmer erfolgen können. Sind alle Teilnehmer angemeldet, kann das Turnier beginnen, welches ebenfalls als Subprozess markiert ist. Nach Beendigung des Turniers werden die Spiele ausgewertet. Nach der Auswertung und dem Speise-und Getränkeverkauf erfolgt die Siegerehrung, womit der Schachtag zu Ende geht. Das entsprechende CTT-Modell ist Abbildung

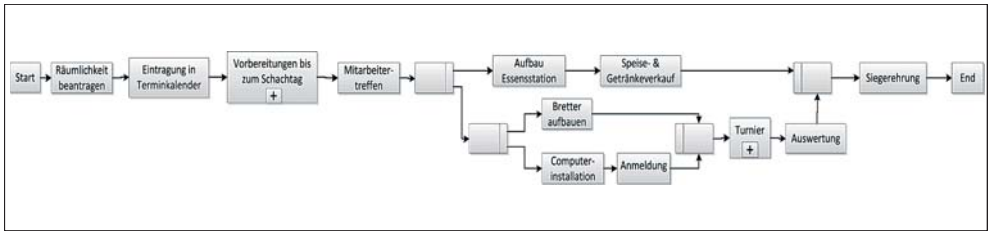


Abbildung 4.1: Gesamtprozess zum Schachtag

4.2 dargestellt. Um eine bessere Vergleichbarkeit zu erhalten, werden in diesem Fallbeispiel wie in ADEPT, auch in CTT die Subprozesse getrennt dargestellt, die

normalerweise als Verzweigungen im eigentlichen CTT-Modell vorkommen würden. In unserem Vergleichsansatz sind die entsprechenden Subprozesse im CTT-Modell ebenfalls mit einem “+”-Zeichen versehen. Im Folgenden betrachten wir den

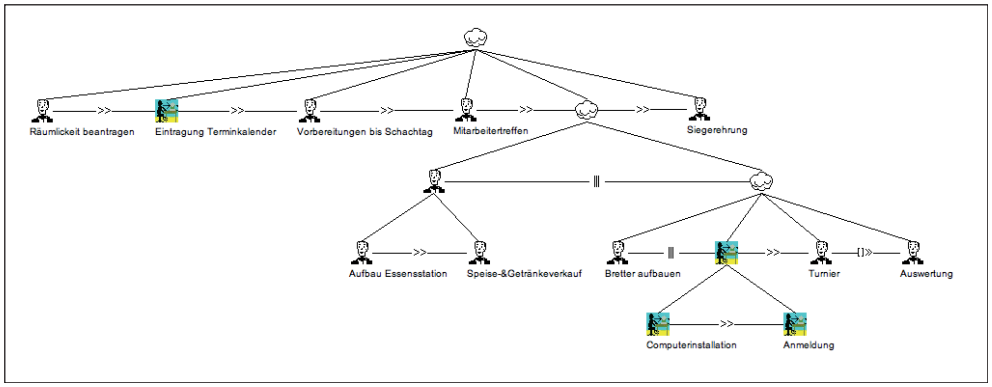


Abbildung 4.2: Gesamtprozess zum Schachtag

Subprozess *Vorbereitungen bis zum Schachtag*, der die zu erledigenden Aufgaben, wie das Einkaufen der Preise (s. Abb. 4.3 a), für den Schachtag beschreibt. Diese sind innerhalb eines AND-Blocks angeordnet und sind somit gleichzeitig auszuführen. Welche Aufgaben insbesondere zu einer Vorbereitung gehören, sind in der Abbildung 4.3 abgebildet.

Der Teilbaum des CTT-Modells, welcher den Subprozess präsentiert, ist in Abbildung 4.3 b) zu sehen. Im Gegenteil zu ADEPT werden die sequentiellen Abläufe, die sich innerhalb des parallelen Blocks befinden, auf einer separaten Blattebene platziert. Diese Anordnung der Knoten hat einen wesentlichen Vorteil zur Folge: Sie erleichtert die Übersichtlichkeit und die Verständlichkeit des Prozessmodells. Der zweite Subprozess *Turnier* beinhaltet die Wiederholung von Aktivitäten, in die-

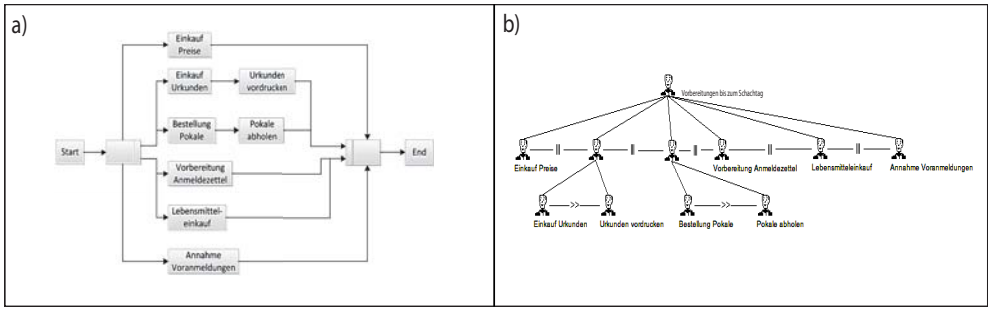


Abbildung 4.3: “Vorbereitungen bis zum Schachtag”-Subprozess

sem Beispiel werden die Spiele mehrmals durchgespielt, da jeder Teilnehmer Runde für Runde einen anderen Gegner hat. ADEPT regelt diese Art der Ausführung von Aufgaben mit Hilfe von Loop-Elementen (Loop-Start Ls, Loop-End Le), die die jeweiligen Knoten umgeben (s. Abb. 4.4 a). Eine elegante Lösung bietet CTT, wie

schon in Kapitel 2 vorgestellt, durch den *-Operator, welcher die Wiederholung symbolisiert (s. Abb. 4.4 b). Die Iteration der kompletten Sequenz auf der 2. Blattebene wird durch einen einzigen *-Operator am Vaterknoten ermöglicht. Bringt man die Iteration an den Blättern selbst an, so wird die Schleife um die Sequenz aufgehoben und die Blätter können einzeln wiederholt werden.

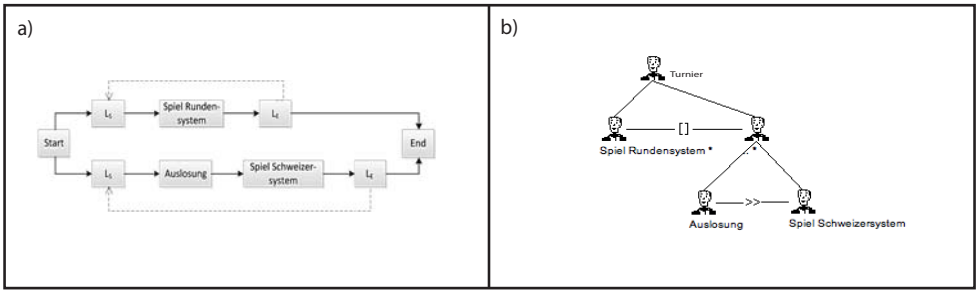


Abbildung 4.4: “Turnier”-Subprozess

Das ADEPT-Modell bietet eine übersichtliche Darstellung der einzelnen Aufgaben, sowie ihrer Abläufe. Zum Problem können die verschachtelten AND-Blöcke werden, die sequentielle Aufgaben mit einem weiteren AND-Block parallelisieren. Innerhalb der verschachtelten AND-Knoten laufen wieder Sequenzen ab. Derartige Verschachtelungen von Verzweigungen erschweren es, die Ausführungsreihenfolge der einzelnen Aufgaben zu verstehen und im Auge zu behalten.
Dieses Problem umgeht CTT, indem die seriellen und parallelen Abläufe auf unterschiedlichen Blattebenen abgebildet werden und somit das Niveau der Verständlichkeit erhöht wird. Doch wie bereits erwähnt, kann sich diese positive Eigenschaft zu einer negativen entwickeln(s. Abschn. 3.4). Die Verteilung der spezifizierten Aufgaben auf verschiedene Verzweigungen bzw. Blattebenen kann bei größeren CTT-Modellen zu Übersichtlichkeitsproblemen und dadurch auch zu Missverständnisse führen.
Wir wollen im weiteren Verlauf der Fallstudie zwei Lösungsansätze für diese Art von Herausforderungen einführen und auf ihre Eignung, sowie auf ihre Nützlichkeit analysieren.
Wie wir schon festgestellt haben, besteht ein CTT-Modell in der Regel aus mehreren Blattebenen. In unserem ersten Lösungsansatz betrachten wir CTT-Modelle mit geringem Verzweigungsgrad. Dies sind CTT-Modelle, mit einer kleinen Anzahl (>= 4) an Ebenen. Dabei spielt es keine Rolle, wie viele Knoten sich auf einer Ebene befinden.
Im ersten Lösungsansatz zeigen wir die Möglichkeit, das modellierte CTT-Modell Ebene für Ebene “aufzuklappen” und so die verschiedenen Perspektiven auf den Baum zu erhalten. Die Vaterknoten, an denen eine Verzweigung auf eine weite-

re Ebene erfolgt, werden mit einem “+” markiert. Bei einem Klick auf diesen wird der Baum um eine Blattebene aufgespannt. Um sich orientieren zu können, wie viele Ebenen noch nicht aufgeklappt sind, wird die Anzahl der noch aufklappbaren Ebenen vor das “+” gesetzt. Mit einem Klick auf den aufgeklappten Vaterknoten, an dem sich das “+” zu einem “-” verändert hat, kann die darunterliegende Ebene bzw. Ebenen wieder geschlossen werden.
In der Abbildung 4.5 haben wir diesen Ansatz am Beispiel des Schachtag-Prozesses umgesetzt. Die Ansicht auf die verschiedenen Ebenen erleichtert dem

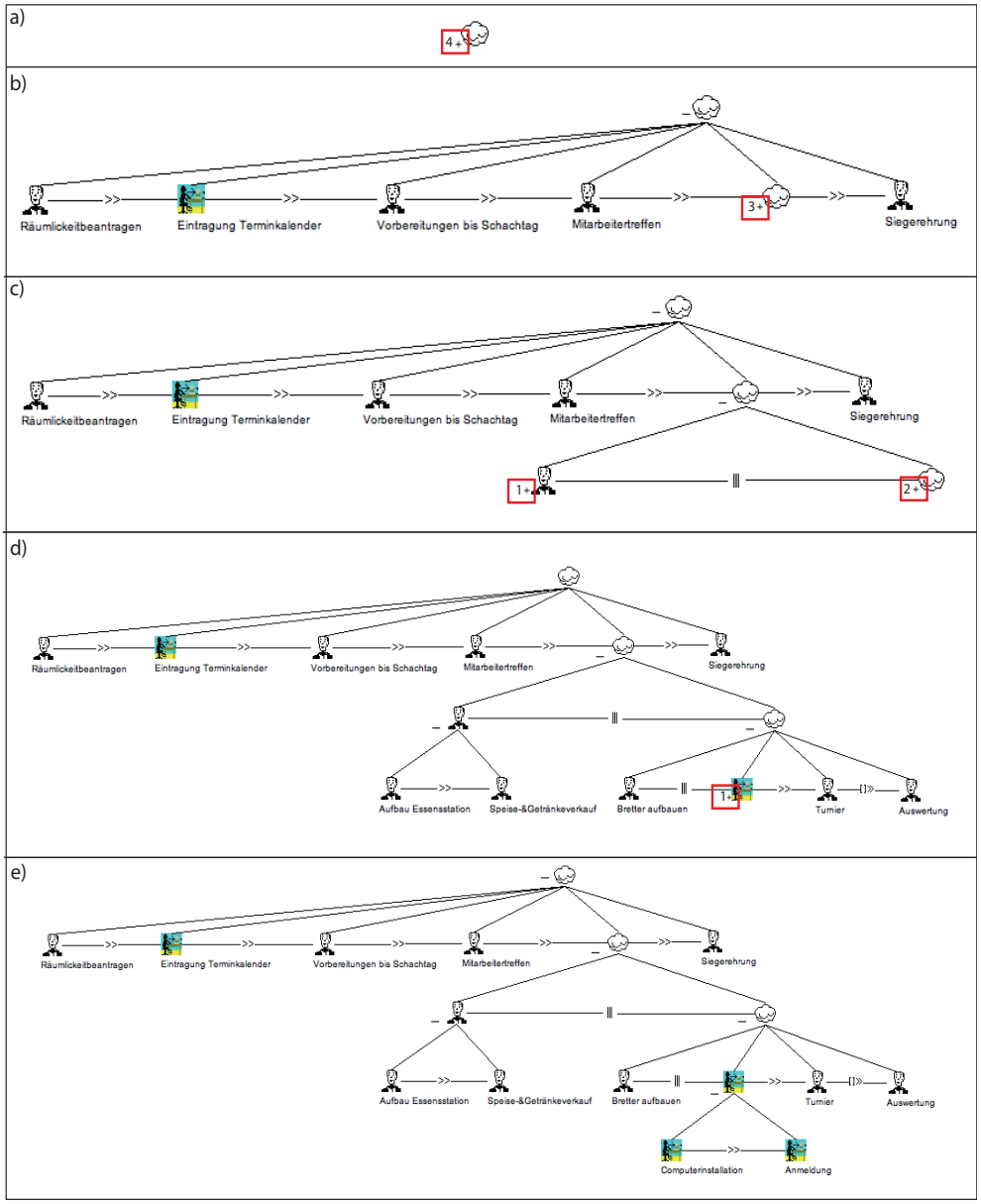


Abbildung 4.5: Darstellung der einzelnen Ebenen

Modellierer Schritt für Schritt für jede Blattebene ein Verständnis aufzubauen. Somit wird der Zusammenhang zwischen den unterschiedlichen Ebenen besser hergestellt, wodurch sich der Grad der Komplexität reduziert.
Das CTTE bietet bereits eine ähnliche Funktion, doch im Unterschied zu unserem Ansatz werden nicht die einzelnen Ebenen, sondern immer der komplette Teilbaum eines Vaterknotens aufgespannt. Ein kleines Beispiel hierzu ist in Abbildung 4.6 a) zu sehen.
Wir wollen diese bestehende Funktionalität des CTTE durch die Kombination mit unserem ersten Lösungsansatz erweitern. Dieser neue Lösungsansatz beschreibt das Vertiefen der einzelnen Ebenen an einem Teilbaum eines Vaterknotens.
Die Abbildung präsentiert unsere “kombinierte” Lösung, in der der Teilbaum des jeweiligen Vaterknotens immer nur um eine Ebene tiefer angezeigt wird.

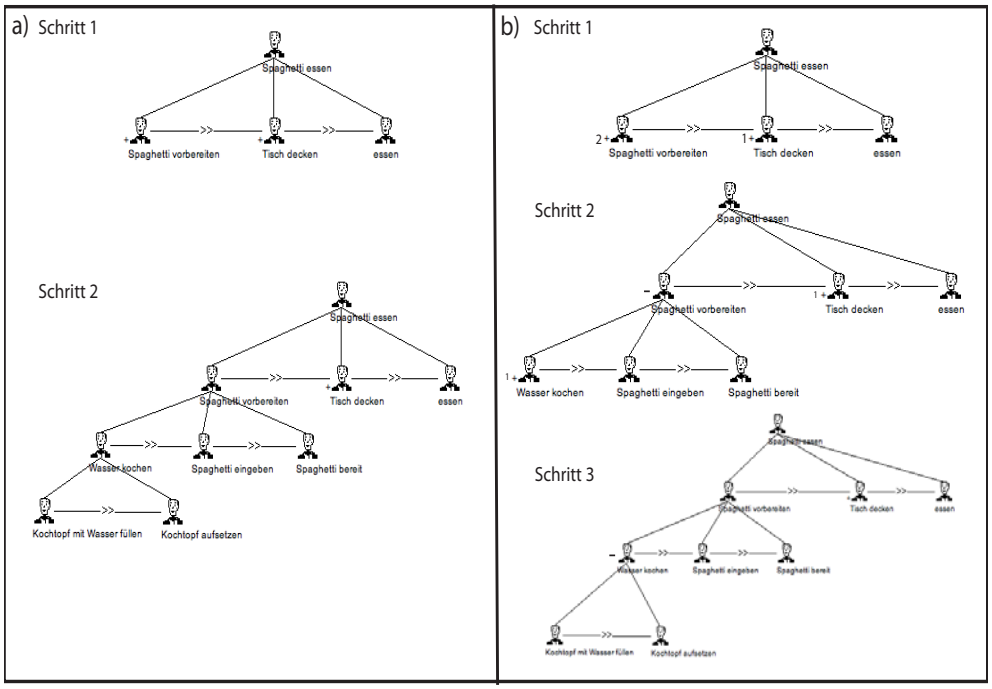


Abbildung 4.6: Darstellung der einzelnen Blattebenen

Dieser Ansatz ist für CTT-Bäume mit hohem Verzweigungsgrad (Anzahl der Blattebenen > 4) sehr gut geeignet, da die einzelnen Schritte der Spezifizierung der Aufgaben besser nachvollziehbar sind und durch die Verknüpfung des oben vorgestellten “Ebenen-Ansatzes” die zusammenhängenden Verbindungen zwischen den verschiedenen Blattebenen und Teilbäumen ersichtlicher werden.

5 Zusammenfassung und Ausblick

Web-Anwendungen nehmen zunehmend einen wichtigen und großen Teil im Leben eines jeden Menschens ein, sei es am Arbeitsplatz, in der Schul- und Berufsausbildung, oder einfach nur in der Freizeitgestaltung. Ein Benutzer erwartet eine möglichst einfach bedienbare Anwendung und ein Entwickler erwartet eine möglichst einfach erstellbare und erweiterbare Anwendung [ref]. Dabei ist der Benutzer mit seinen Bedürfnissen, Interessen, Zielen, Aufgaben und seiner Umgebung, der Maßstab für die Entwicklung des Produktes.
Mit der steigenden Wichtigkeit von Anwendungen, nimmt die nutzerzentrierte Gestaltung von Anwendungen einen immer größeren Stellenwert ein und ist zukünftig nicht mehr wegzudenken. Die graphische Notation CTT bildet dabei eine entscheidende Grundlage für die aktive Benutzerbeteiligung im Softwareentwicklungsprozess.
Die Arbeit befasst sich mit der Erstellung von Prozessmodellen mittels CTT. Dieser Entwicklungsprozess ist ein sehr mühsames Geschäft, insbesondere dann, wenn es bei der Erstellung um eine sehr detaillierte Modellierung geht. CTT mit seiner baumartigen Struktur unterstützt die Entwicklung von präzisen und spezifischen Modellen. Aus fachlicher Sicht und für ein grundlegendes Prozessverständnis genügt oftmals eine relativ grobe Darstellung der prinzipiellen Prozessschritte, wie sie im Normalfall ablaufen. Schwierigkeiten bei großen CTT-Modellen treten auf, wenn zunehmend Details und die Behandlung von Sonderfällen, sowie nachträgliche Änderungen hinzukommen. Diese lenken unter Umständen eher vom Hauptanliegen des Prozessmodells ab und erschweren das Verständnis. Andererseits sind auch detaillierte Darstellungen notwendig, beispielsweise als genaue Vorgabe für Entwickler, die daraus System- und Benutzeraufgaben ableiten und diese mit in die eigentliche Systementwicklung einbringen können. Durch die Anwendung der Modellierungsumgebung CTTE wird die Erstellung derartiger Detailmodelle ermöglicht, aber es ist dennoch sehr aufwändig, sogar bei kleineren Modellen, Aufgaben bis ins Detail zu spezifizieren. Obwohl CTTE bereits viele verschiedene Tool-Unterstützungen bietet, schützt es nicht vor semantischen Fehlern.
Die vorgestellten Lösungsansätze in der Fallstudie sind hilfreich, um Nachteile des CTT beseitigen zu können, oder sogar zu noch besseren Modellierungsergebnissen führen könnten. In Zukunft sind derartige Erweiterungen sehr essentiell be-

züglich der Verständlichkeit und Gebrauchbarkeit, somit sind sie folglich wichtig für die immer mehr wachsende Nutzerzentrierung in Gestaltungsprozessen.

Literaturverzeichnis

[1] Maria Francesca Costabile, Daniela Fogli, Piero Mussio, and Antonio Piccinno. A Meta-Design Approach to End-User Development. *VLHCC*, pages 11–13, 2005.

[2] Christian Beutenmüller. Deklarative XML-Sprachen für Benutzerschnittstellen. Master's thesis, Universität Leipzig, 2005.

[3] Sebastian Feuerstack, Marco Blumendorf, and Sahin Albayrak. Bridging the Gap between Model and Design of User Interfaces. In *Proccedings of Informatik 2006 Conference Dresden Germany*. Springer, 2006.

[4] Fabio Paternò, C Mancini, and S Meniconi. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In S Howard, J Hammond, and G Lindgaard, editors, *Proceedings of the IFIP TC13 Interantional Conference on HumanComputer Interaction*, volume 96 of *IFIP Conference Proceedings*, pages 362–369. Citeseer, Citeseer, 1997.

[5] Task Models, Models As, and A Bridge Between. Task models and system models as a bridge between hci and software engineering. *Human-Computer Interaction*, pages 357–385, 2009.

[6] G. Zachmann. `http://zach.in.tu-clausthal.de/teaching/info2_06/folien/03_baeume_1_4up.pdf`, zuletzt besucht am 17.05.2011.

[7] Fabio Paternò. ConcurTaskTrees : An Engineered Approach to Model-based Design of Interactive Systems. *The Handbook of Analysis for HumanComputer Interaction*, pages 1–18, 2002.

[8] G Mori, F Paterno, and C Santoro. CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Transactions on Software Engineering*, 28(8):797–813, 2002.

[9] ConcurTaskTrees Environment. `http://giove.isti.cnr.it/tools/CTTE/home`, zuletzt besucht am 22.05.2011.

[10] B Weber, M Reichert, and S Rinderle-Ma. Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Data & Knowledge Engineering*, 66(3):438–466, 2008.

[11] E Gamma, R Helm, R Johnson, and J Vlissides. *Design patterns: elements of reusable object-oriented software*, volume 206 of *Addison-Wesley Professional Computing Series*. Addison-Wesley, 1995.

[12] Barbara Weber, Stefanie Rinderle, and Manfred Reichert. *Change Patterns and Change Support Features in Process-Aware Information Systems*, volume 4495 of *Lecture Notes in Computer Science*, pages 574–588. Springer, 2007.

[13] Elmar Braig. Unterstützung von Zeitaspekten in Workflow-Management-Systemen - Anforderung, Lösungsansätze und offene Fragestellungen. Master's thesis, Universität Ulm, 2004.

Name: Yasemin Agbulak Matrikelnummer: 656813

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Name