Spezifikation zusätzlicher Leistungsmerkmale für ein Telefonvermittlungssystem

Thorsten Michels

31. Juli 1997

Universität Kaiserslautern
AG Rechnernetze
Betreuer: Jan Bredereke

Zusammenfassung

Das Modell des Intelligenten Netzwerks ist eine Abstraktion von Telefonvermittlungssystemen und beschreibt auch deren Erweiterungen. Zunächst wird ein einfaches Basissystem spezifiziert, das dann um weitere Leistungsmerkmale, sog. Features, erweitert wird. Im Rahmen dieser Arbeit haben wir ein bereits bestehendes, in Estelle spezifiziertes Basissystem um sechs Features erweitert. Dabei konnten wir verschiedene Stile für die Featurespezifikation in Estelle überprüfen. Wir entwerfen Prinzipien für eine verhaltenerhaltende Transformation, die geeignete Ansatzpunkte für neue Features schaffen kann. Für das Ergänzen von neuen Rufnummern haben wir eine einfache Methode entwickelt. Wir zeigen zwei Schwächen von Estelle beim Erweitern von Systemen auf. Schließlich berichten wir über unsere Erfahrungen mit dem im IN-Modell verwendeten Prinzip der Detection Points.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Bei dem Entwurf von Kommunikationssystemen, wie z. B. Telefonvermittlungssystemen, ist es vorteilhaft, wenn man zuerst einen Basisdienst definiert und diesen dann um zusätzliche Leistungsmerkmale, sog. Features, erweitert. Der Kerndienst kann dann bereits verifiziert werden, bevor Erweiterungen hinzugefügt werden. Die Trennung führt darüberhinaus zu kleineren, überschaubaren Einheiten, so daß eine Erweiterung deutlich vereinfacht wird. Zum Beispiel können Kunden bei (amerikanischen) Telefonsystemen heute schon aus einer großen Anzahl von Features auswählen, die sie einzeln zusätzlich kaufen können. Mittlerweile sind an die tausend Features beschrieben worden.
Diese Features können sich auch gegenseitig stören, und da es so viele sind, ist es unmöglich, alle Kombinationen vollständig zu testen. Diese „Feature-Interaktionen“ sind ein ernstes Problem. Mit dem Einsatz einer „Formalen Beschreibungstechnik“ (Formal Description Technique, FDT), wird es möglich, das Problem auf einer abstrakteren Ebene zu behandeln. Eine solche FDT ist Estelle. Estelle basiert auf dem Konzept der (kommunizierenden erweiterten) endlichen Automaten. Die Untersuchungen in der AG Rechnernetze zielen darauf, wie man ein Kommunikationssystem aus einem Kerndienst und vielen Erweiterungen derart in Estelle spezifizieren kann, daß die Interaktionen zwischen den Erweiterungen möglichst früh erkannt werden können oder eventuell auch gar nicht erst auftreten.

In einer vorangegangenen Projektarbeit [Ill95] wurde ein stark vereinfachtes Basis-Telefonvermittlungssystem in Estelle spezifiziert. Das Vermittlungssystem basiert auf den Standards für das Diensteintegrierende Digitale Netzwerk (ISDN, [ITU93a]) und für die Intelligenten Netzwerke (IN, [DuVi92]). Letztere beschäftigen sich mit den Möglichkeiten, wie Vermittlungssysteme für neue Leistungsmerkmale geöffnet werden können. Aufgabe dieser Arbeit war es, dem Basissystem diverse Features hinzuzufügen. An diesen Features sollen dann Methoden zur Erkennung von Feature-Interaktionen erprobt werden, insbesondere automatische Werkzeuge wie z. B. das Tool „Decfine“, entwickelt in [Bar96]. In einer weiteren parallel laufenden Projektarbeit [Jer97] werden andere Features dem Basissystem hinzugefügt. Durch

dieses parallele Arbeiten an verschiedenen Leistungsmerkmalen soll herausgefunden werden, ob Feature-Interaktionen zwischen Leistungsmerkmalen, die tatsächlich unabhängig voneinander realisiert wurden, gefunden werden.

In Kapitel 2 werden zunächst die Grundprinzipien der Intelligenten Netzwerke (IN) dargestellt und das Basissystem vorgestellt. Weiterhin werden einige Anforderungen an den Spezifikationsstil für die Erweiterungen des Basissystems erläutert. In Kapitel 3 stellen wir Entwurfsregeln für das nachträgliche Ändern des Basissystems vor. Im Laufe der Arbeit hat es sich ergeben, daß es für das Spezifizieren der Features und die Untersuchung der Interaktionen sinnvoll sein kann, Änderungen am Basissystem durchzuführen. Damit diese nicht zu Konflikten mit den bereits spezifizierten Teilen führen, haben wir diese Regeln entworfen. Die durchgeführten Änderungen und die sechs spezifizierten Features werden in Kapitel 4 beschrieben. In Kapitel 5 berichten wir über unsere Erfahrungen mit der Feature-Spezifikation in Estelle und dem Prinzip der „Detection Points“ der Intelligenten Netzwerke. Kapitel 6 faßt die Ergebnisse dieser Arbeit zusammen.

Kapitel 2

Grundlagen und Begriffserklärungen

2.1 Das Intelligente Netzwerk

Das Modell des Intelligenten Netzwerks (IN) [DuVi92] beschreibt eine Abstraktion und Erweiterung von (Telefon-)Vermittlungssystemen. Basierend auf bereits existierenden Vermittlungssystemen sollen die Standards, die von der ITU-T[1] entwickelt werden [ITU93e], für alle Arten von Netzwerken Verwendung finden: öffentliche Netzwerke, ISDN, Datennetze, mobile Netze u. ä.
Zu den Zielen des Intelligenten Netzwerks zählen:

- Erhöhung der Service-Einführungsgeschwindigkeit, um eine schnelle, vom Markt angetriebene Einführung neuer Dienste zu ermöglichen;
- Erweiterung der Art der Dienste über die traditionelle Sprach- und Datenübertragung hinaus, z. B. mit Informationsdiensten, Breitband- und Multimediaunterstützung;
- ein korrektes und konsistentes Funktionieren der Dienste bei jedem Anbieter und über Anbietergrenzen hinweg;
- Unterstützung der Einführung neuer Technologien und ihr Zusammenwirken mit den alten, da diese nicht über Nacht ausgetauscht werden können.

Dazu wurde ein Architekturkonzept für INs entwickelt, dessen Charakteristiken in folgenden Punkten zusammengefaßt werden können:

- ausgedehnter Gebrauch von Techniken der Informationsübertragung und von Netzwerkressourcen;
- Integration von Diensterfindung und -implementierung durch wiederverwendbare, standardisierte Netzwerkfunktionen;

[1] International Telecommunication Union - Telecommunication Standardization Sector, früher CCITT

- flexible Zuweisung modularisierter, wiederverwendbarer Netzwerkfunktionen auf Hardwareeinheiten durch dienstunabhängige Schnittstellen;
- Zugriff der Dienstanbieter auf den Prozeß der Dienstezusammenstellung durch Kombination von Netzwerkfunktionen;
- Benutzerkontrolle über die benutzerspezifischen Dienstmerkmale;
- standardisierte Verwaltung der Dienste und ihrer Funktionen.

Für den Entwurf des Intelligenten Netzwerks wurde ein Begriffsmodell definiert, das aus vier Ebenen besteht, die unterschiedliche Abstraktionsniveaus des IN darstellen:

– Die Dienstebene (service plane) repräsentiert eine ausschließlich dienstorientierte Sicht auf das IN für den Dienstnutzer und den Diensterbringer. Sie enthält keine Angaben über Implementierung der Netzwerkdienste.

– Die globale Funktionsebene (global functional plane) bietet die Sicht auf die verschiedenen Funktionen des IN. Sie enthält das Modell für die Anrufbearbeitung und die dienstunabhängigen Funktionen (service independent building blocks). Das Netzwerk wird als eine Einheit betrachtet.

– Die verteilte Funktionsebene (distributed functional plane) zeigt die Verteilung der Funktionen im IN. Hier werden die funktionalen Objekte (functional entities, FE), ihre Aktionen (FEA) und die Beziehungen und Informationsflüsse untereinander beschrieben.

– Auf der physikalischen Ebene (physical plane) werden die physikalischen Aspekte beschrieben: die Geräte, welche funktionalen Objekte sie realisieren, und die Protokolle, mit denen sie kommunizieren.

Für diese Arbeit sind vor allem die verteilte Funktionsebene und ihre funktionalen Objekte wichtig. Die wichtigsten Objekte sind die folgenden:

– Die Call Control Agent Function (CCAF) stellt den Zugang des Benutzers zum Netzwerk bereit und dient als Interface zwischen ihm und den Anrufsteuerungsfunktionen.

– Die Call Control Function (CCF) ist für die Anrufübertragung und deren Steuerung zuständig.

– Die Service Switching Function (SSF) stellt eine Menge von Funktionen bereit, die für die Wechselwirkungen von CCF und SCF benötigt werden.

– Die Service Control Function (SCF) steuert die Bearbeitung von Dienstaktivitäten durch das IN oder den Benutzer. Nach Bedarf wendet sich die SCF dazu an SDF oder SRF.

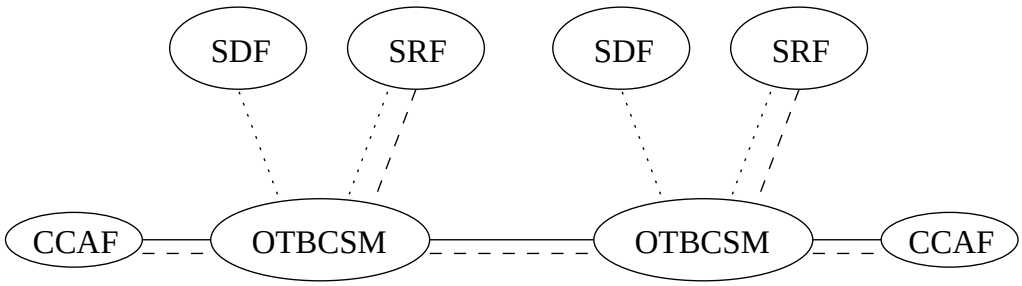
– Die Service Data Function (SDF) steuert den Zugriff auf dienstbezogene und Netzwerkdaten durch die SCF und verwaltet diese Daten auch.

– Die Specialized Resources Function (SRF) stellt spezielle Ressourcen (z. B. Spracherkennungsmöglichkeiten, Konferenzbrücken usw.) zur Verfügung, die zur Ausführung der vom IN bereitgestellten Dienste benötigt werden.

Im IN-Konzept besteht das Modell der Anrufbearbeitung (Basic Call Model, BCM) zunächst aus zwei Teilmodellen, jeweils eines für den Anrufer (Call Originator) und eines für den Angerufenen (Call Terminator). Sie setzen sich zusammen aus einer Menge von Zuständen (oder auch „Points In Call“, PIC), einer Menge von sog. „Detection Points“ (DP) und Übergängen zwischen ihnen. Detection Points werden benutzt, um die verschiedenen Stationen innerhalb der Anrufbearbeitung des BCM anzuzeigen. Sie können aktiviert werden, um einem Dienst anzuzeigen, wenn der DP erreicht wird, damit der Dienst das Basismodell an dieser Stelle erweitern kann. In den einzelnen PICs findet die Verarbeitung der Schritte des Anrufs statt: vom Abheben des Hörers über die Anrufvermittlung bis zum Auflegen des Hörers.
Das Basissystem wird um sog. „Capability Sets“, Mengen von Features, erweitert. Diese Capability Sets werden, teilweise oder komplett, inkrementell hinzugefügt. Zur Zeit ist das erste Set, CS-1, mit einer ganzen Reihe von Features fest definiert (siehe [DuVi92], [ITU93c]). Hauptsächlich betreffen die Features in CS-1 den Verbindungsauf- und -abbau und sind nur auf einen Kunden und ein Verbindungsende ausgerichtet. CS-2 ist noch in Vorbereitung.

2.2 Das Grundsystem

In [Ill95] wurde ein einfaches Anrufbearbeitungsmodell gemäß dem BCM in Estelle spezifiziert und implementiert. Das System unterstützt genau zwei Benutzer, die in einer statischen Verbindung miteinander stehen. Zur Vereinfachung wurden CCF, SSF und SCF zu einem Modul zusammengelegt, dem OTBCSM (Originating and Terminating Basic Call State Model). Jedem dieser beiden Module ist ein CCAF-Modul zugeordnet. Diese Systemarchitektur ist in Abbildung 2.1 zu sehen.
Die Benutzerschnittstelle wurde mit dem graphischen Frontend xtpanel implementiert. Abbildung 2.2 zeigt das Interface im Betrieb.



Kanalarten:

- – – – – Sprache
- ——— Signalisierung
- ····· Steuerung

OTBCSM	Originating & Terminating Basic Call State Model
CCAF	Call Control Agent Function
SDF	Service Data Function
SRF	Specialized Resources Function

Abbildung 2.1: Die vereinfachte Systemarchitektur

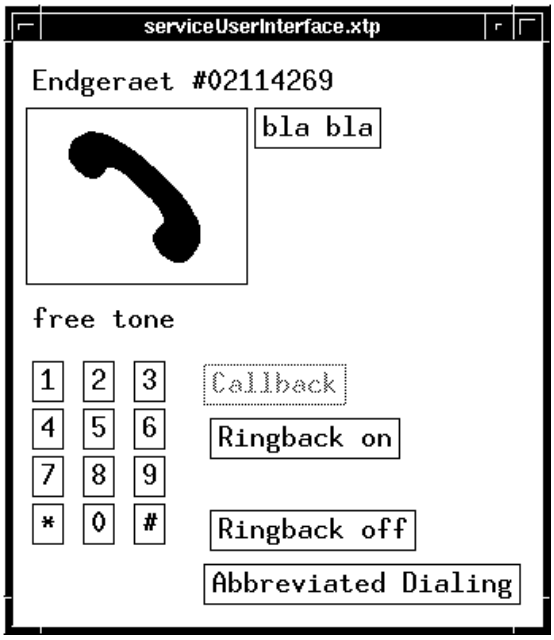


Abbildung 2.2: Die **Benutzerschnittstelle**, sozusagen der Telefonapparat.
Ganz oben die Rufnummer. Darunter der „abgenommene“ Hörer und der Button für die Übermittlung einer `blabla`-Nachricht (Simulation natürlicher Sprache). Unter dem Hörersymbol werden Klingel- und Signaltöne ausgegeben, hier das Freizeichen. Rechts die zusätzlichen Tasten für die Features. Das Callback-Feature war während des Screenshots nicht aktiviert, deshalb ist seine Taste im „non-sensitive“-Modus. Zwischen den Ringback-Tasten ist Platz für Nachrichten des CCBS-Features.

Außerdem sind bereits die Grundlagen und Verbindungen für die Module SDF und SRF angelegt. Die Vereinfachungen, insbesondere die statische Anlage, waren notwendig, um Decfine die Untersuchung des Systems zu erleichtern.

Die Spezifikation des OTBCSM-Moduls geschah im wesentlichen nach dem ITU-T Standard Q. 1214 [ITU93d]. Die daraus entnommenen Zustandsgraphen finden sich in Abbildung 2.3. Bei der Umsetzung wurden sowohl PICs als auch DPs als Estelle-Hauptzustände und die Übergänge als Estelle-Transitionen umgesetzt. Aufgrund der statischen Anlage des Systems ergaben sich vor allem folgende Änderungen gegenüber Q. 1214: Die PICs O_Null und T_Null wurden zu einem neuem PIC Null zusammengelegt, von dem aus in die entsprechenden Nachfolgezustände des Originating oder Terminating BCSM gesprungen wird. Dadurch kann allerdings nicht mehr gleichzeitig ein Anruf durch das OBCSM ausgeführt werden und einem anderen Anrufer ein Besetzt-Zeichen durch das TBCSM übermittelt werden. Für diesen Fall wurde dem Modul ein Subautomat hinzugefügt, der das Absenden der Besetzt-Nachricht übernimmt. Die Realisierung geschah in Anlehnung an [Bre95b] durch Variablen, die als Subzustandsvariablen verwendet wurden.

Da die Übermittlung der natürlichen Sprache der beiden Gesprächsteilnehmer für unsere Zwecke nicht interessant ist, wird über die Sprachkanäle nur eine `blabla`-Nachricht übertragen. Es ist aber einfach, diese Kanäle um zusätzliche Nachrichten zu erweitern, wenn dies für neue Features benötigt wird, z. B. mündliches Angeben der Rufnummer anstatt Drücken von Zifferntasten.

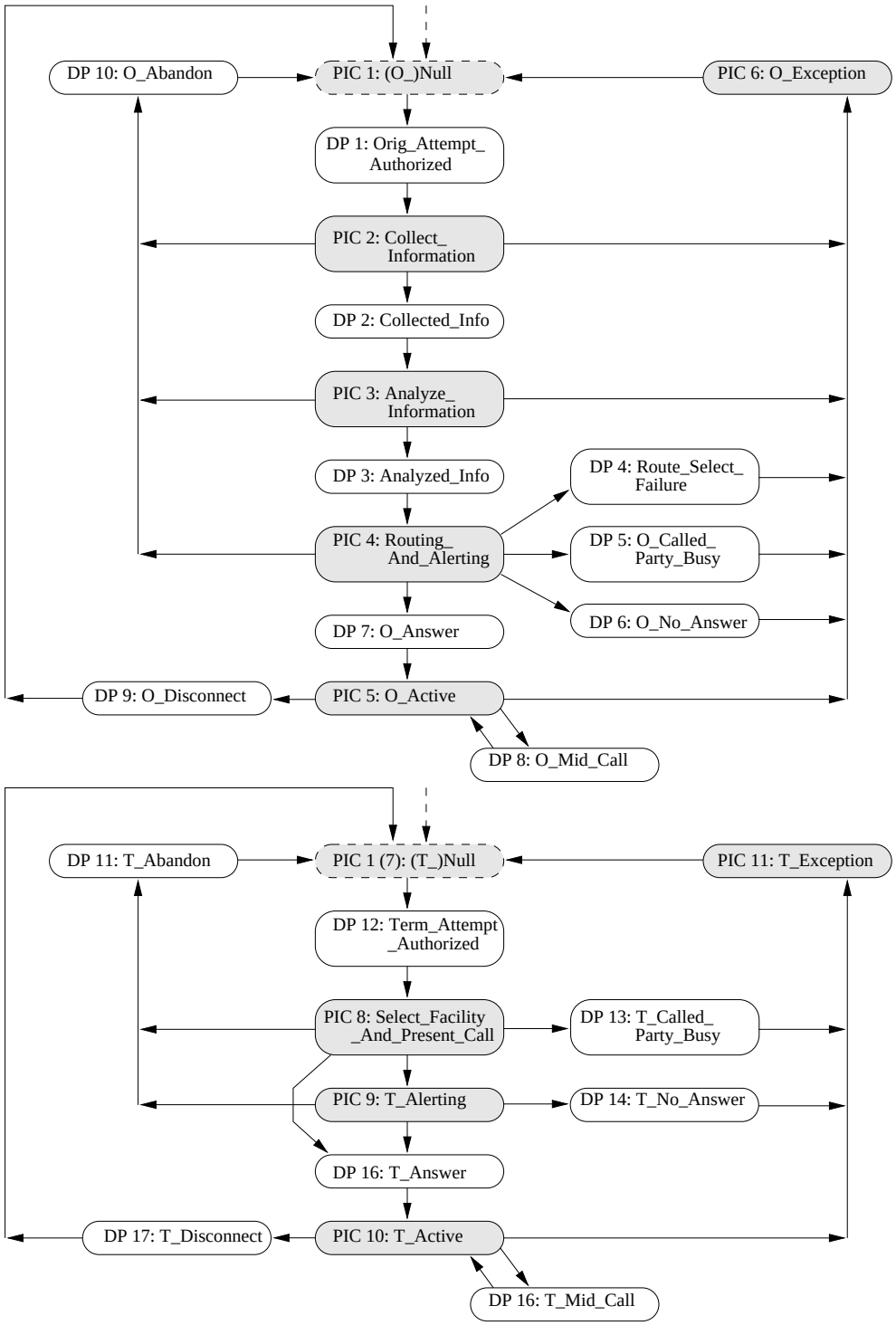


Abbildung 2.3: Der Ablaufplan eines einfachen Telefongesprächs: Oben aus Sicht der Anruferseite (Originating Basic Call Model, OBCM), unten aus Sicht des Angerufenen (Terminating Basic Call model, TBCM). Die Hauptzustände (PICs) sind grau unterlegt, die Detection Points weiß.

2.3 Spezifikationsstil für Erweiterungen

2.3.1 Grundregeln

Um die erweiterte Spezifikation auf Interaktionen zwischen den neuen Features untersuchen zu können, gibt es verschiedene Stilregeln für die Spezifikation, die wir hier kurz zusammenfassen (siehe [Bre97], [Bre95a], [Ill95]).

- Alle Features müssen getrennt voneinander spezifiziert werden. Jeder Schritt einer Ausführungsfolge (in Estelle also jede Transition) muß eindeutig einem Feature zuzuordnen sein. In Estelle geschieht dies durch die Vergabe von eigenen Prioritäten(-konstanten) für jedes Feature.
- Ein Feature darf dem System nur etwas hinzufügen. Bereits bestehende Teile dürfen nicht gelöscht oder geändert werden.
- Das System darf nur grobkörnig verändert werden. Das heißt, daß nur komplette Transitionen hinzugefügt werden dürfen, nicht z. B. einzelne Zeilen im Transitionsrumpf oder zusätzliche Bedingungen in Provided-Klauseln.
- Analog dürfen auch Zustandskomponenten wie Variablen oder Module nur eingefügt, nicht gelöscht werden. Wenn der Wertebereich einer Zustandskomponente geändert werden muß, darf er nur ergänzt werden.

Altes Verhalten, das nicht mehr benötigt wird, wenn Features dem System fest hinzugefügt werden, muß während des Erweiterungsprozesses erhalten bleiben und darf erst nach der Analyse auf Interaktionen entfernt werden.

2.3.2 Erweiterungen an Detection Points

Das Basic Call State Model sieht vor, daß die Features an den Detection Points aktiviert werden: Wenn der Automat in der Ausführung einen Detection Point erreicht, wird dies angezeigt und das Feature kann an dieser Stelle ansetzen. In Estelle wurden sowohl DPs als auch PICs mit Hauptzuständen spezifiziert. Dadurch ergeben sich für die Features zwei Möglichkeiten, um einzusetzen:

1. Die Transition vom DP zum folgenden PIC wird durch eine neue Transition des Features oder eine Folge von Transitionen ersetzt. Sie erhalten eine höhere Priorität als die Transitionen des Basissystems und verdecken diese somit. Nachteil: Dadurch kann beim Erreichen des DPs immer nur jeweils ein Feature aktiviert werden. Dies kann aber auch gewollt sein, z. B. wenn ein Feature einen PIC komplett ersetzen muß. In unserer Spezifikation war dies im Feature LongNumber nötig (siehe den ersten Punkt auf Seite 22).

2. Das Feature setzt am DP an und kehrt nach Durchführung seiner Aktionen wieder an den gleichen DP zurück. Dadurch können mehrere Features an diesem DP ansetzen, was im Sinne eines defensiven Spezifikationsstils zu begrüßen ist. Weiterhin können durch Festlegung unterschiedlicher Prioritäten für die Features unterschiedliche Ausführungsreihenfolgen oder auch Nichtdeterminismus erreicht werden. Nachteil: Damit das System nicht in eine Endlosschleife gerät, weil das Feature immer wieder aktiviert wird, ist pro Feature eine Boolean-Subzustandsvariable notwendig, die bei Aktivierung auf TRUE gesetzt und bei passender Gelegenheit wieder auf FALSE zurückgesetzt wird. Dadurch wird der Zustandsraum aufgebläht. Die Untersuchung durch die Tools Confine und Decfine wird Aufschlüsse geben, wie störend dies tatsächlich ist.

2.3.3 Zustandsverfeinerungen

Manche Features (oder auch Funktionen des Basissystems) müssen ihre Aufgaben in einem konkreten Hauptzustand durchführen, benötigen aber in ihrem Ablauf mehrere Teilzustände, z. B. um einzelne Teilschritte des Ablaufs voneinander zu trennen oder um sich Werte oder Ereignisse zu „merken“. Der Hauptzustand muß also in diese Teilzustände aufgegliedert werden. In [Bre95b] wurden zwei grundsätzliche Methoden beschrieben, wie Zustände in Estelle verfeinert werden können. Wir möchten diese Methoden anhand eines Beispiels mit einem Miniautomaten mit zwei Hauptzuständen (`idle` und `busy`) erläutern, von denen der zweite in drei Subzustände (`busy_entry`, `busy_internal` und `busy_exit`) verfeinert werden soll.

1. Verfeinerung durch Namenskonvention

   In dieser Methode werden die drei Subzustände als Estelle-Hauptzustände implementiert, die `busy` ersetzen.

```
STATE idle,
      {busy :} busy_entry, busy_internal, busy_exit;
INITIALIZE TO idle
  BEGIN END;
TRANS FROM idle TO busy_entry
  BEGIN {...} END;
TRANS FROM busy_entry TO busy_internal
  BEGIN {...} END;
TRANS FROM busy_internal TO busy_exit
  BEGIN {...} END;
TRANS FROM busy_exit TO idle
  BEGIN {...} END;
```

Ein Feature, das auf diese Weise spezifiziert wird, wird also in seinem Standardablauf normalerweise nicht unterbrochen. Es führt seine Aufgaben in den Subzuständen nacheinander aus. Es ist nicht vorgesehen, daß parallel dazu andere Aktionen stattfinden. Nach der Aktivierung durchläuft es seine verschiedenen Zustände und kehrt dann zum Basissystem zurück.

2. Verfeinerung durch einen Subautomaten

   Hierbei werden die Subzustände nicht durch Estelle-Hauptzustände, sondern durch eine eigene Variable dargestellt. Der Hauptzustand bleibt in seiner Form erhalten, der erweiterte Zustand wird um den Wert der Subzustandsvariable ergänzt.

```
STATE idle, busy;
TYPE busySubstatesType = (busy_entry, busy_internal, busy_exit);
VAR busySubstate: busySubstatesType;
INITIALIZE TO idle
  BEGIN busySubstate:=busy_entry END;
TRANS FROM idle TO busy
  BEGIN {...} END;
TRANS FROM busy TO SAME
  PROVIDED busySubstate=busy_entry
  BEGIN {...} busySubstate:=busy_internal END;
TRANS FROM busy TO SAME
  PROVIDED busySubstate=busy_internal
  BEGIN {...} busySubstate:=busy_exit END;
TRANS FROM busy TO idle
  PROVIDED busySubstate=busy_exit
  BEGIN {...} busySubstate:=busy_entry END;
```

   Dies ist von der Funktionalität her identisch zum ersten Verfahren.

2.3.4 Wahlweises Hinzufügen von Features

Alle Teile der Featurespezifikation werden in Präprozessoranweisungen (ähnlich wie in der Programmiersprache C) geschachtelt:

```
#ifdef Featurename
  {Spezifikation von Featurename}
#endif Featurename
```

Dies erlaubt ein wahlweises Hinzunehmen oder Weglassen der Features beim Compilieren. Es ist darauf zu achten, daß Features, die auf anderen Features aufbauen, dies auch in den Präprozessoranweisungen tun. In unserer Spezifikation benutzte z. B. das Feature Personal Numbering (PN) das Feature LongNumber:

```
#ifdef LongNumber
#ifdef PN
  {Spezifikation von Personal Numbering}
#endif PN
#endif LongNumber
```

Kapitel 3

Prinzipien für nachträgliche Änderungen am Basissystem

Im Laufe der Arbeiten an den Features hat sich herausgestellt, daß es nicht immer sinnvoll ist, die Features nur durch Ergänzungen des Basissystems zu spezifizieren. Im konkreten Fall ergab es sich, daß zwei ansonsten voneinander unabhängige Features nicht in der Lage waren, an geeigneten getrennten Punkten im Basissystem anzusetzen, da das Basissystem nicht fein genug spezifiziert war. Die Features setzten an derselben Transition an, wollten aber verschiedene Teile der Transition verändern. In unserem Fall sollte das Feature PN etwas am Kopf einer Transition verändern, indem es weitere Möglichkeiten zum Feuern hinzufügt. Das Feature ACB sollte dem Transitionsrumpf etwas hinzufügen.[1] Es ist möglich, diese Interaktion durch ein weiteres Feature aufzulösen, das die entsprechende Transition durch eine neue ersetzt, die die Funktionalität von PN und ACB vereint. Sobald allerdings weitere Features an dieser Transition ansetzen möchten, müßte man dann für jede mögliche Kombination von aktivierten Features ein Auflösungsfeature spezifizieren, was die Spezifikation in unvertretbarer Weise aufbläht. In unserem Fall traf das für gleich zwei weitere Features zu: S0130 und S0180 aus [Jer97]. Der entsprechende Zustand im Basissystem ist so „beliebt“, weil in ihm zusätzliche Rufnummern, z. B. für Service-0130-Dienste, hinzugefügt werden.
Es zeichnete sich also ab, daß es einfacher wäre, das Basissystem und die an dieser Stelle ansetzenden und bereits spezifizierten Features zu ändern, als eine exponentiell steigende Zahl von Auflösungsfeatures zu implementieren. Die Transformation würde aus einer Verfeinerung der entsprechenden Transition bestehen. Dabei muß allerdings folgende Bedingung gelten:

> Das Verhalten nach der Transformation des Systems muß konform [Got95] zum Verhalten vor der Transformation sein. In diesem Fall bedeutet dies, daß das sichtbare Verhalten nach der Transformation spezifischer

[1] Es ging dabei um die Transitionen im Subautomaten, der „Besetzt“-Nachrichten verschickt. Die Namen der Transitionen beginnen mit `Both_try_to_connect`.

als vorher ist.

Wir geben dazu einige Transformationsregeln für die zu verfeinernde Transition an. Ein formaler Nachweis, daß diese Regeln korrekt sind, würde den Rahmen dieser Projektarbeit überschreiten. Wir geben aber einige Gründe für die Wahl dieser Regeln an. In unseren Fällen bedeutete dies, die betroffene Transition in mehrere voneinander unabhängige Transitionen aufzuspalten. Für eine Transition, die von Zustand S_1 in Zustand S_2 führt, sieht dies folgendermaßen aus:

1. Eine Transition mit Ausgangszustand S_1, die die Nachricht entgegennimmt und deren Parameter in Hilfsvariablen ablegt.

2. Falls nötig: Eine oder mehrere Transitionen, die anhand dieser Parameter eine Auswahl treffen oder in verschiedene Subzustände verzweigen.

3. Eine oder mehrere Transitionen, die den Rumpf der alten Transition enthalten, also alle Variablenänderungen vornehmen, OUTPUT-Kommandos ausführen und in Zustand S_2 übergehen.

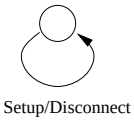
Nachrichten von anderen Automaten werden demnach in denselben Zuständen und damit auch zu den gleichen Zeiten wie vorher entgegengenommen. Eigene Nachrichten werden, wie vorher auch, beim Übergang zu S_2 verschickt. Damit ist das nach außen sichtbare Verhalten des verfeinerten Automaten konform zum Original, solange keine weiteren Nachrichten während der Ausführung der Transitionen 2 und 3 eintreffen. In diesem Fall muß gewährleistet sein, daß diese Nachrichten erst nach Ausführung der dritten Transition entgegengenommen werden.
Auch Subautomaten, die parallel zu den verfeinerten Transitionen arbeiten könnten, dürfen in dieser Zeit nicht ausgeführt werden. Dies ist natürlich eine harte Einschränkung, und wir haben auch noch keine allgemeinen Regeln, wie man diese Subautomaten umformen müßte. Es gibt jedoch Möglichkeiten, diese Einschränkung aufzuweichen. Vor allem muß auf das Ein-/Ausgabeverhalten und auf Datenabhängigkeiten zur geänderten Transition geachtet werden. Dies bedarf aber noch einer genaueren Untersuchung. Im Fall, der in unserer Spezifikation auftrat, haben wir die existierenden Subautomaten „von Hand“ überprüft und festgestellt, daß sie die Ausführung der verfeinerten Transitionen nicht stören.
Für Features, die an der ursprünglichen Transition ansetzen wollten, gilt die gleiche Regel über das Aufspalten: Features, die die empfangene Nachricht ändern, müssen die erste Transition überdecken; Features, die die mitgeschickten Parameter benutzen oder auswerten, müssen nach der ersten Transition ansetzen und u. U. die zweite Transition überdecken; Features, die Variablen verändern oder Nachrichten verschicken, müssen dies nach der zweiten Transition bzw. anstatt der dritten Transition tun, je nachdem, ob sie das alte Verhalten ergänzen oder ersetzen.

Im o. a. Beispiel sollten die Features PN und ACB an derselben Stelle des Features LongNumber ansetzen, so daß eine Verfeinerung dieses Features nötig war.

Genaugenommen stellte bereits LongNumber an dieser Stelle eine Verfeinerung des Basissystems dar (siehe auch Abbildung 3.1). [2] Im Basissystem besteht der Subautomat, der eine Besetzt-Nachricht verschickt, wenn bereits ein Gespräch stattfindet, nur aus einer Transition, die die Setup-Nachricht entgegennimmt und eine Disconnect-Nachricht verschickt. Die ursprüngliche Spezifikation von LongNumber ersetzte diese Transition durch eine, die die Setup-Nachricht entgegennahm und die Parameter speicherte, und durch zwei Transitionen, die aufgrund dieser Parameter ein Disconnect mit Grund „besetzt“ oder „falsche Nummer“ schickten. PN beeinflußte diese beiden Transitionen, weil es eine gültige Nummer hinzufügt. ACB möchte das Verhalten der Besetzt-Transition ändern. Also mußten die beiden Transitionen aufgespalten werden in zwei, die die Unterscheidung treffen, ob die Rufnummer gültig war oder nicht, und zwei, die die entsprechenden Disconnect-Nachrichten verschicken. Damit kann PN an den ersteren ansetzen und ACB an den zweiten, und der Konflikt ist behoben.

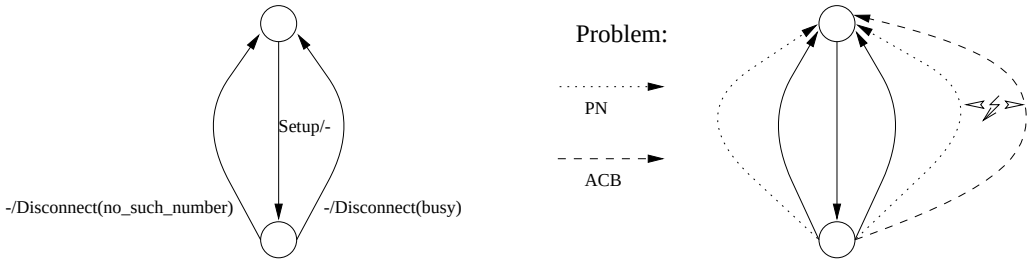
[2] Es handelte sich also nicht um eine nachträgliche Änderung am *Basissystem*, sondern an LongNumber. Das Prinzip ist aber das gleiche.

1. Basissystem



Keine Unterscheidung von verschiedenen Faellen

2. LongNumber, erster Entwurf



3. LongNumber, zweiter Entwurf

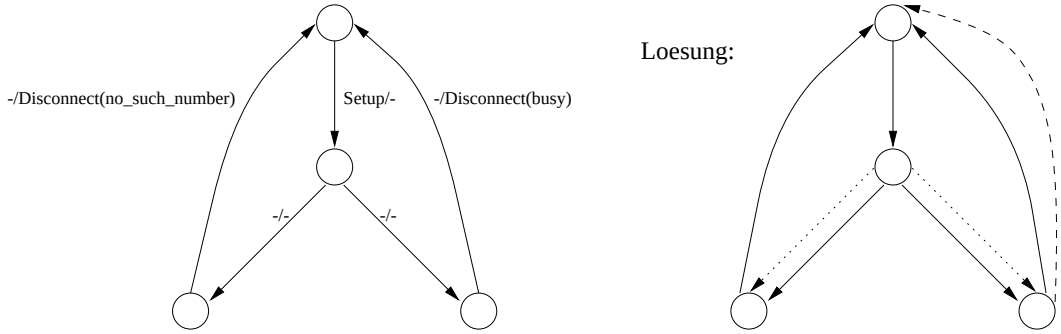


Abbildung 3.1: **Beispiel für nachträgliche Änderungen**

Kapitel 4

Erweiterungen des Basissystems

Dieses Kapitel umfaßt die Änderungen und Erweiterungen, die an der Spezifikation des Basissystems vorgenommen wurden. Im ersten Abschnitt werden die Änderungen aufgezählt, die am Basissystem selbst notwendig waren. In den weiteren Abschnitten werden die neu spezifizierten Features beschrieben.

4.1 Änderungen am Basissystem

In diesem Abschnitt werden Änderungen beschrieben, die aus verschiedenen Gründen am Basissystem vorgenommen wurden. Es handelt sich hierbei nicht um Features, die ergänzt wurden, oder die im vorigen Kapitel beschriebenen korrektheitserhaltenden Verfeinerungen. Es sind Änderungen in der Funktionalität des Basissystems, die notwendig waren, um die Features zu spezifizieren oder die Untersuchung durch Decfine zu erleichtern. Sie wurden also auch nicht in dem in Abschnitt 2.3.4 beschriebenen Spezifikationsstil (mittels `#ifdef`-Erweiterungen) durchgeführt, sondern ersetzen die alte Spezifikation. Wir haben diese Änderungen vor der endgültigen Spezifikation der Features und der Analyse vorgenommen.

4.1.1 Delay-Transitionen

Das Originalbasissystem enthielt an mehreren Stellen Delay-Transitionen, die das Klingeln des Telefons oder verschiedene Töne, wie z. B. das Besetztzeichen, spezifizierten. Die Transition `RepeatedRingTone` z. B. verschickte in einem konstanten Abstand Nachrichten an das CCAF-Modul, die dieses veranlaßten, das Telefon klingeln zu lassen. Diese Delay-Transitionen verursachten bei der Erreichbarkeitsanalyse durch Decfine einen besonders hohen Verzweigungsgrad im Zustandsgraphen: Die Delay-Transitionen erzeugten einen Zyklus aus mehreren Schritten im Ablauf des OTBCSM. Für jeden Schritt in diesem Zyklus muß wegen des unsynchronisierten Interleavings ein eigener Teilbaum aus Folgezuständen aufgebaut werden. Dies führte zu einer unerwünschten und vermeidbaren Zustandsexplosion.

Wir waren der Meinung, daß die Ausgabe der einzelnen Klingeltöne nicht Sache des OTBCSM, sondern des CCAF ist. Das OTBCSM sollte nur noch mitteilen, wann das Intervall, in dem geklingelt werden soll, beginnt und wann es aufhört. Die Funktionalität des Basissystems bleibt damit gleich, die Spezifikation und Umsetzung des Endgeräts kann sogar flexibler gehandhabt werden. Dazu verschickt das Basissystem die Nachricht nur noch einmal zu Beginn des Intervalls und setzt die neu eingeführte Subzustandsvariable `Sending_Tone` auf TRUE. Am Ende des Intervalls wird `Sending_Tone` zurückgesetzt und eine `Clear`-Nachricht an das CCAF geschickt. Sie bewirkt das Ende *aller* Tonausgaben. Wir sind davon ausgegangen, daß das CCAF nie zwei Töne gleichzeitig von sich geben soll, weil dies der meist verwendeten Hardware entspricht. Dies erspart die Einführung jeweils einer eigenen Intervallende-Nachricht pro möglichen Ton.
Nach dieser Änderung bleiben im Basissystem nur noch fünf Delay-Transitionen. Sie sind für den Abbruch der Verbindung notwendig, wenn nach einer bestimmten Zeitspanne kein Kontakt hergestellt wurde. Damit ist auch jeweils ein Zustandswechsel verbunden. Diese Transitionen können also jeweils nur einmal schalten und verursachen dadurch nicht eine so große Zustandsexplosion wie die ersetzten.

4.1.2 Absenderidentifizierung

Das Basissystem ist nur auf zwei Benutzer ausgelegt. Deshalb war es am Anfang nicht nötig, den Absender einer Nachricht zwischen zwei OTBCSMs zu identifizieren. Dies änderte sich, als für die einzelnen Benutzer mehrere Rufnummern möglich wurden (z. B. im Feature PN), und wird natürlich auch bei einem Ausbau des Systems auf mehrere Anschlüsse wichtig werden.
Deshalb wurde in allen Nachrichten, die von OTBCSM zu OTBCSM geschickt werden, die Nummer des Absenders ergänzt. [ITU93a] sieht einen Call-Reference-Parameter vor, der mit einem komplizierterem Algorithmus das aktuelle Gespräch eindeutig identifiziert. Für unsere Zwecke reicht die Überprüfung der Absendernummer aber völlig aus.
Außerdem wurde durch den neuen Parameter jeweils eine Transition pro Nachricht zur Beseitigung von unspezifiziertem Empfang notwendig.

4.1.3 Präzedenzmatrix

Ursprünglich wurden in der Spezifikation auch alle Konstanten für die Prioritäten des Basissystems und der Features festgelegt. Wir haben sie in ein eigenes File (`bcs-gen-prec.e`) ausgelagert, das mit `#include` eingelesen wird. Dieses File kann bequem mit dem Tool genprio erzeugt werden. genprio erzeugt aus einer Präzedenzmatrix oder -liste automatisch Konstantendefinitionen in Estelle-Quellcode.

4.1.4 Variablenreset in PIC_Null

Zur Untersuchung des Systems wird ein globaler Nullzustand benötigt, in dem sich das System in Ruhe befindet. Dieser Nullzustand definiert Anfang und Ende für die Analyse. Bei der Rückkehr in den Ruhezustand müssen dann auch alle Variablen denselben Wert wie beim Start der Analyse haben. Für einige Variablen wurde das im Originalsystem noch nicht durchgeführt. Dazu wurde jetzt die Transition `PIC_Null_Reset_Variables` eingeführt.

4.2 LongNumber

Das LongNumber-Feature war das erste Feature, das dem Basissystem hinzugefügt wurde. Gleichzeitig ist es auch das bisher umfangreichste. Dafür bauen aber auch alle weiteren Features, die bisher spezifiziert wurden, darauf auf, indem sie die Erweiterungen durch LongNumber nutzen. Das heißt aber auch, daß sie nur mit LongNumber benutzt werden können.
Das Basissystem sieht nur zwei Teilnehmer vor. Deshalb wurden auch die Rufnummern und der Numbering-Plan besonders einfach gehalten: Für die beiden Teilnehmer werden nur die beiden Rufnummern „1“ und „2“ vergeben. Rufnummern, die aus mehr als einer Ziffer bestehen, sind nicht vorgesehen. Dies wird von LongNumber erweitert: Die Rufnummern haben nun 4+4 Ziffern. Die ersten vier Ziffern sind für die Bereichsvorwahl bestimmt, die immer mit der Ziffer 0 beginnen soll. Sie können auch als Kennung für andere Features, z. B. „Service 0130“ genutzt werden. Die zweiten vier Ziffern werden dann für die eigentliche Rufnummer oder Durchwahl benutzt. Der Gesprächspartner kann sowohl mit als auch ohne die Vorwahl angerufen werden. Den OTBCSMs soll die Liste der Vorwahlnummern und die Liste der Rufnummern mit der eigenen Vorwahl bekannt sein. Dadurch ergeben sich zwei Szenarien:

1. Der Benutzer ruft jemanden in einem anderen Bereich an und gibt dazu eine achtstellige Nummer ein. Das OTBCSM überprüft dann die Vorwahl. Wenn sie gültig ist, wird der Anruf an das entsprechende OTBCSM weitergegeben, wo die Durchwahl überprüft wird. Sollte einer der beiden Tests fehlschlagen, wird dem Anrufer ein entsprechendes Zeichen gegeben.

2. Der Benutzer ruft jemanden in seinem eigenen Bereich an und wählt dazu dessen vierstellige Durchwahl. Diese vier Ziffern werden sofort vom OTBCSM des Anrufers überprüft. Dann wird der Anruf an den Partner weitergegeben.[1]

Folgende Änderungen wurden durchgeführt:

[1] Der tatsächliche Unterschied besteht also zur Zeit nur darin, daß die Rufnummer an verschiedenen Stellen überprüft werden. An der Zuordnung von CCAF zu OTBCSM ändert sich nichts. Das liegt an der statischen Struktur des Systems.

- PIC 3 (`Analyze_Information`) wurde komplett neu geschrieben.
  In diesem Zustand wird überprüft, ob die eingegebene Telefonnummer schon komplett ist. Dabei feuert die entsprechende Transition des Basissystems bereits nach einer eingegebenen Ziffer. Der Benutzer hat dann keine Möglichkeit die weiteren sieben Ziffern noch einzugeben. Da sich diese Transition auch nicht abschalten oder überdecken läßt, mußte der ganze Zustand kopiert und neu geschrieben werden (Siehe dazu auch Abschnitt 5. 1. 3). Der neue Zustand heißt PIC 3a `PIC_Analyze_Information_LongNumber`.

- Die `Setup`-Nachricht, die dem Angerufenen den Verbindungswunsch mitteilt, wurde durch eine neue Nachricht ersetzt, die auch die angerufene Nummer als Parameter enthält. Dies ist notwendig, da der Angerufene in Szenario 1 die Gültigkeit dieser Nummer überprüfen muß.

- Weil diese Gültigkeit bisher nur beim Absender getestet wurde, ergibt sich dadurch auch ein neuer Grund, einen Verbindungsaufbau abzulehnen. Auch die `Disconnect`-Nachricht wurde durch eine neue ersetzt, die einen Parameter mit dem Grund der Ablehnung mit sich führt. Für diesen Parameter wurde ein neuer Variablentyp erklärt (`DisconnectCauseType`), der sich leicht um weitere Gründe erweitern läßt, wenn dies einmal nötig ist. Zur Zeit enthält er die Möglichkeiten „`busy`“ und „`no_such_number`“.

- Aus demselben Grund mußten die beiden Teile von PIC 7 (`T_Null_And_Authorize_Termination_Attempt`) in zwei neue PICs mit den entsprechenden Funktionalitäten aufgespalten werden. Dies geschah in Übereinstimmung mit [ITU93b], abgesehen von der Numerierung. In `T_Null` (PIC 7) wird wie bisher auch der Verbindungswunsch entgegengenommen. In `Authorize_Termination_Attempt` (PIC 7a) wird festgestellt, ob die Durchwahlnummer gültig ist und ggf. ein `Disconnect` geschickt.
  Dadurch ergeben sich auch Änderungen in den Detection Points, die ebenfalls in Übereinstimmung mit [ITU93b] durchgeführt wurden: DP 12 (`DP_Term_Attempt_Authorized`) wird abgeschafft und ersetzt durch DP 12a (`DP_Term_Attempt`) für PIC 7a, DP 12b (`DP_Term_Authorized`) für PIC 8, sowie DP 19 (`DP_Term_Denied`) für den Übergang von PIC 7a zu PIC 11 (`PIC_T_Exception`).

- Eine ähnliche Situation ergab sich für den Subautomaten, der Verbindungswünsche beantwortet, wenn bereits eine Verbindung besteht. Alle PICs, die im vorherigen Punkt angesprochen wurden, sollten in diesem Subautomaten noch einmal auftauchen. Deshalb wurde die eine Transition, die all dies vorher erledigen konnte (`Both_try_to_connect_Send_Busy`) in verschiedene Einzelschritte aufgespalten, die mit Hilfe der Statusvariable `Both_Connect_Status` nacheinander abgearbeitet werden:
  Wenn `Both_Connect_Status` den Wert `no_second_call` hat, kann die

`Setup`-Nachricht entgegengenommen werden. Ihre Parameter werden in dafür neu eingerichteten Variablen zwischengespeichert. In `got_second_call` werden die Parameter ausgewertet. Je nachdem ist der nächste Subzustand dann `valid_nr_second_call` oder `not_valid_nr_second_call`. Dort wird dann die `Disconnect`-Nachricht mit dem Parameter „`busy`“ oder „`no_such_number`“ verschickt.
Daß die Aufteilung in drei Einzelschritte nötig ist, war zu Beginn der Spezifikation von LongNumber nicht klar; es hat sich erst im Laufe der Arbeit an den Features PN und ACB herausgestellt. Zu Anfang wurden der zweite und dritte Schritt in einer Transition bearbeitet. Dadurch kam es zu einer Interaktion der Features PN und ACB, die während des Spezifizierens entdeckt wurde und aufgelöst werden konnte, indem diese Transition aufgespalten wurde. Dies geschah nach den Richtlinien aus Kapitel 3.

4.3 Customized Ringing (CRG)

[ITU93c], der Standard, in dem die Features beschrieben sind, faßt sich sehr kurz in diesen Beschreibungen. Zu CRG wird nur erwähnt, daß es dem Benutzer erlaubt, für eine Liste möglicher Anrufer unterschiedliche Klingeltöne einzurichten.
Wir haben uns entschieden, dieses Feature so einzurichten, daß Anrufer, die nur eine vierstellige Nummer benutzt haben, einen anderen Klingelton bekommen. Man kann dies auch als eine Kennzeichnung für hausinterne Anrufe betrachten. Die Unterscheidung wird also nur anhand der ersten Ziffer der gewählten Nummer gemacht: Wenn sie 0 ist, ist es ein „normaler“ Anruf, sonst einer, der ein besonderes Klingelzeichen verursachen soll.
Dazu mußte nur in `PIC_T_Alerting_SS_Call_Received` die Transition `Ring_Tone` überlagert werden.[2]

4.4 Personal Numbering (PN)

Auch dieses Feature führt einen neuen Klingelton ein. Hiermit kann sich ein Benutzer für einen Endapparat mehrere Rufnummern erteilen lassen, die dann unterschiedliche Töne hevorrufen, z. B. für private und geschäftliche Anrufe oder für verschiedene Familienmitglieder.
In unserer Implementierung haben wir uns dafür entschieden, den beiden Benutzern jeweils eine zusätzliche Nummer zu erteilen, die als letzte Ziffer eine 0 hat, in den anderen Ziffern aber mit der normalen Nummer identisch ist. Diese Nummer erzeugt dann einen anderen Klingelton.

[2] Vor der Ersetzung der Delay-Transitionen für die Ausgabe der Töne (siehe Abschnitt 4.1.1) mußte allerdings der komplette Subzustand ersetzt werden, da die Delay-Transition nicht komplett überlagert oder abgeschaltet werden kann (vergl. Abschnitt 5.1.3).

Die notwendigen Änderungen für den Klingelton sind analog zum CRG-Feature. Für die zusätzlichen Rufnummern sind eine neue Funktion, die diese testet, und Transitionen, die sie dem Numbering-Plan hinzufügen, nötig. Für Details zu dieser Methode siehe Abschnitt 5.1.5.

4.5 Automatic Callback (ACB)

Die Aktivierung von ACB erlaubt es dem Benutzer, den letzten Anrufer zurückzurufen, der versucht hat, ihn anzurufen. Dabei muß sich das System bei jedem Verbindungsaufbauwunsch die Rufnummer des Anrufers merken. Dazu wurde die Variable `Last_number_ACB` angelegt. Nachdem festgestellt wurde, daß ein Verbindungsaufbauwunsch tatsächlich für den Benutzer bestimmt war, wird die Nummer des Anrufers dort gespeichert. Das geschieht im Normalfall in `DP_Term_Authorized`. Wenn zur Zeit des Verbindungsaufbauwunsches bereits ein Gespräch läuft, geschieht es im Subautomat `Both_try_to_connect` im Subzustand `valid_nr_second_call`.
Der Benutzer kann danach, anstatt eine Rufnummer zu wählen, dieses Feature aufrufen. Für unser System haben wir dabei folgende **Stilfestlegung** getroffen:

> Für Features, die vom Benutzer aufgerufen werden können, wird im Userinterface eine eigene Taste eingerichtet. Das CCAF kann dann diese Taste in beliebige Codes und Nachrichten an das OTBCSM umsetzen, falls das nötig ist. Dadurch ersparen wir uns die aufwendige Spezifikation, die sich ergibt, wenn Features über die normale Zehnertastatur aktiviert werden müssen, weil keine zusätzlichen Tasten verfügbar sind.[3] Falls keine Hardware mit zusätzlichen Tasten verfügbar sein sollte, könnte ein separates Modul Ziffernkombinationen in die geforderten Nachrichten umsetzen, damit die eigentliche Spezifikation unbelastet bleibt. Dieses separate Modul ist nicht Teil der Spezifikation.

In `PIC_Analyze_Information_LongNumber` kann das Feature also aufgerufen werden. Dann wird `DestinationPhoneNumber` der Wert von `Last_Number_ACB` zugewiesen, und der Zustand wird sofort zum nächsten DP verlassen. Außerdem wird die Subzustandsvariable `ACB_Activation` auf TRUE gesetzt. Nach erfolgreichem Verbindungsaufbau in `DP_O_Answer` wird damit erkannt, daß `Last_Number_ACB` zurückgesetzt werden kann. Dadurch wird die Rückkehr in den globalen Nullzustand ermöglicht, in dem `Last_Number_ACB` zurückgesetzt ist. In diesem Fall oder bei erfolglosem Verbindungsaufbau wird dann `ACB_Activation` wieder auf FALSE gesetzt. Wenn ACB aktiviert wurde, ohne daß eine Rufnummer gespeichert wurde, wird eine Fehlermeldung an das CCAF geschickt.

[3] Im Telefonsystem des Autors muß z. B. das Feature „Anklopfen“ durch die Tastenfolge `*43#` aktiviert werden.

4.6 Abbreviated Dialling (ABD)

Hierbei handelt es sich um das Kurzwahl-Feature. Der Benutzer kann sich für längere Rufnummern eine kürzere Tastenkombination programmieren. In unserem Fall kann man das mit bis zu zehn Nummern tun, die dann mittels Drücken der neuen Kurzwahltaste, gefolgt von einer entsprechenden Zifferntaste, abgerufen werden. Die Nummern werden bei aufgelegtem Hörer einprogrammiert. Dazu werden erst die Kurzwahltaste und die Ziffer für die Kurzwahl gedrückt. Danach wird die Rufnummer eingegeben, abgeschlossen wiederum durch die Kurzwahltaste.
Die Programmierung erfolgt in `PIC_Null`. Durch Drücken der ABD-Taste gelangt man in den Subzustand `PIC_Null_ABD_GetIndex`. Dann drückt man die Zifferntaste, mit der man die lange Nummer abkürzen und aufrufen will, und kommt in den Subzustand `PIC_Null_ABD_GetPhoneNumber`. Wenn * oder # gedrückt wurden, gelangt man in `PIC_Null` zurück und eine Fehlermeldung wird an den Benutzer verschickt. Im zweiten Subzustand werden die Tastendrücke für die Rufnummer entgegengenommen. Dabei können auch * und # benutzt werden. Mit der ABD-Taste wird dann in `PIC_Null` zurückgekehrt. Für die zu speichernden Rufnummern wurde ein Array eingerichtet. Der Benutzer kann die Programmierung auch jederzeit abbrechen, indem er den Hörer abhebt.
Das Abrufen der Nummern erfolgt dann nach Abheben des Hörers in `PIC_Analyze_Information_LongNumber`. Nach Drücken der ABD-Taste gelangt das System in einen neuen Zustand `PIC_Analyze_Information_ABD`, in dem der nächste Zifferndruck abgewartet wird. Je nach gedrückter Ziffer wird dann `DestinationPhoneNumber` ein Wert aus dem Array zugewiesen und die Anrufbearbeitung normal fortgesetzt. Drücken von * oder # produziert die gleiche Fehlermeldung wie eine nichtexistente Rufnummer.

4.7 Completion of Calls to Busy Subscriber (CCBS)

[ITU93c] faßt sich auch hier wieder sehr kurz: Falls die angerufene Nummer besetzt war, wird der Anrufer informiert, sobald die Nummer wieder frei wird. [UniKL97] ist etwas detaillierter, was Benutzung und Ablauf dieses Features betrifft.
In Abbildung 4.1 ist als Überblick ein Time-Sequence-Diagramm für den normalen Ablauf des Features gegeben.
Im User-Interface wurden zwei Tasten für Start und Stop des Features implementiert sowie ein eigenes Anzeigefeld für Nachrichten des Features. Wenn der angerufene Anschluß besetzt ist und das OTBCSM sich im Zustand `PIC_O_Exception_Busy` befinden, hat der Benutzer die Möglichkeit, CCBS zu starten. Das System schickt dann eine `CCBS_Request`-Nachricht an den Partner. Der schickt ein `CCBS_Ack` mit einem Boolean-Parameter zurück, der TRUE ist, falls die Benachrichtigung erfolgen kann, wenn der Anschluß wieder frei wird. Wenn er bereits einen anderen Benutzer benachrichtigen soll, setzt er den Parameter auf FALSE. Im CCBS-Anzeigefeld wird

dann ein Text angezeigt, daß das Feature nicht benutzt werden kann. Eine erfolgreiche Bestätigung wird ebenfalls dort angezeigt. Der Benutzer muß bis zu einer Anzeige warten, bevor er den Hörer auflegt, andernfalls wird das Feature abgebrochen.
Danach kann der Benutzer sein Gerät wie sonst auch behandeln. Wenn der Angerufene sein Gespräch beendet und den Hörer aufgelegt hat, sendet sein OTBCSM ein `CCBS_Answer`. Hat der Benutzer seinen Hörer aufgelegt, sendet das System ein Klingelsignal an das Interface (`CCBS_Ring`). Nimmt der Benutzer dann den Hörer auf, versucht das System sofort einen neuen Verbindungsaufbau: Es springt in `PIC_Routing_and_Alerting` und sendet eine `Setup`-Nachricht. Sollte aus irgendeinem Grund der Partner besetzt sein, wird CCBS automatisch wieder aktiviert. Der Benutzer kann den Ablauf des Features jederzeit durch die CCBS-Stopp-Taste abbrechen. Außerdem bricht das Feature automatisch nach einer gewissen Zeitspanne ab, wenn der Benutzer nicht auf die Benachrichtigung reagiert, daß der Angerufene jetzt bereit ist.

Das Feature wurde in einem Subautomaten spezifiziert, da die meisten Schritte parallel zum normalen System ablaufen. Dazu gibt es zwei Subzustandsvariablen: `CCBS_State_Sender` für den Ablauf beim wartenden Benutzer und `CCBS_State_Receiver` für den reagierenden Benutzer. Der Ruhezustand für beide Variablen ist `Inactive`, der Sender durchläuft nacheinander `Wait_for_Ack`, `Request_Sent` und `Answer_Received`, der Empfänger kommt in den Zustand `Activated`.
Zur Spezifikation des Features gehören des weiteren eine Transition, die die zur Speicherung der Rufnummer verwendete Variable zurücksetzt, um wieder in einen globalen Nullzustand zu gelangen, sowie diverse Transitionen, die unspezifizierten Empfang vermeiden.

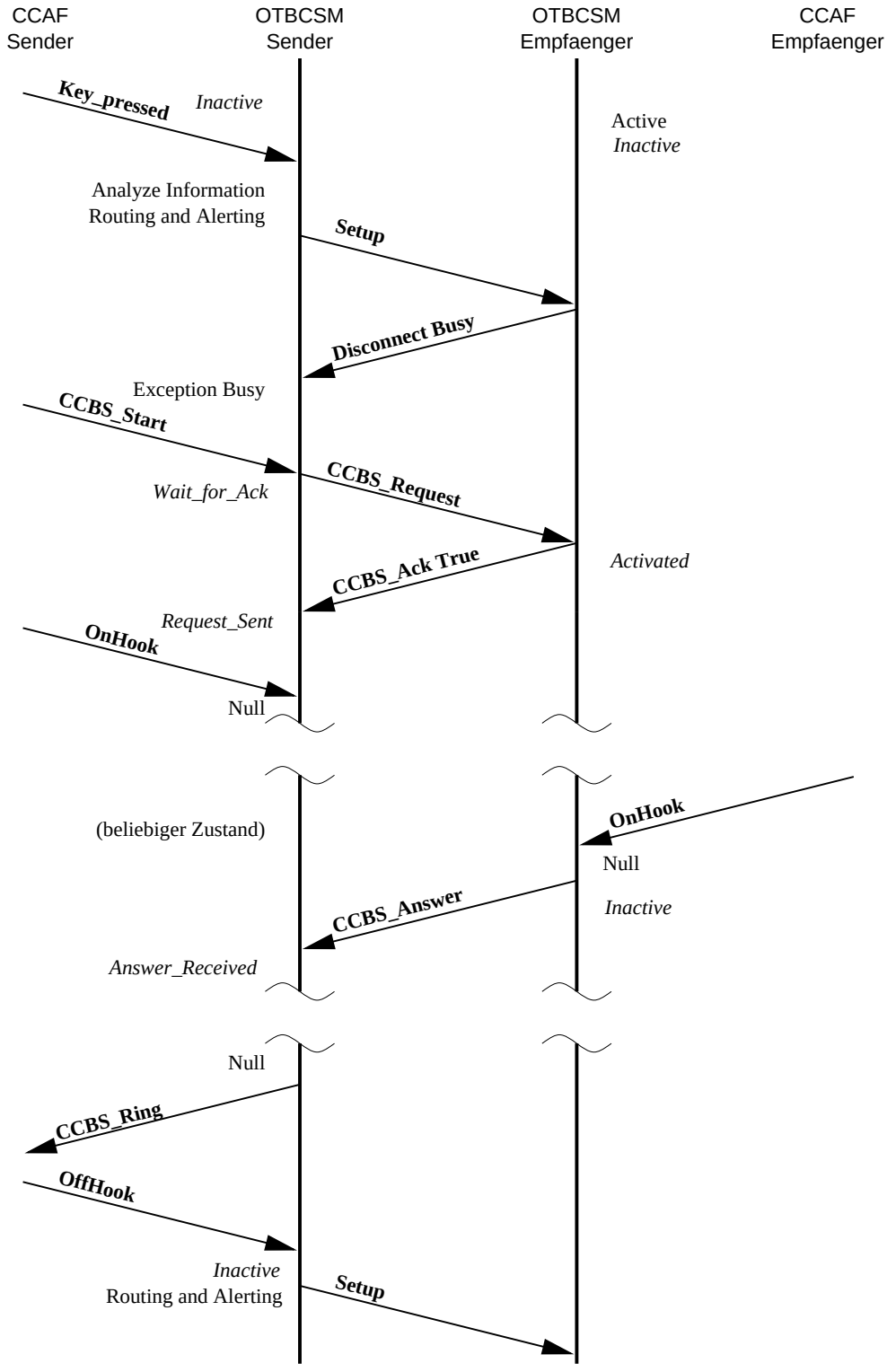


Abbildung 4.1: **Standardablauf des CCBS-Features.** In normaler Schrift der Hauptzustand, in *Kursivschrift* der Zustand des Subautomaten für CCBS, in **Fettdruck** die Nachrichten, die verschickt werden.

Kapitel 5

Ergebnisse

5.1 Feature-Spezifikation mit Estelle

5.1.1 Zustandsverfeinerungen

In Abschnitt 2.3.3 haben wir zwei Methoden vorgestellt, wie Features unter Verwendung von Zustandsverfeinerungen spezifiziert werden können: durch Estelle-Hauptzustände mit einer Namenskonvention bzw. durch Verwendung einer Subzustandsvariable.
Wenn eine Zustandsverfeinerung nötig war und das Feature seine Aufgabe sequentiell erledigen konnte, haben wir uns für die Methode der Namenskonvention entschieden. Sie wurde bereits in [Ill95] für die Spezifikation des Basissystems verwendet. Unserer Meinung nach ist sie übersichtlicher und leichter nachvollziehbar. Gegenüber der zweiten Methode hat sie außerdem den Vorteil, daß sie keine zusätzliche Zustandvariable einführen muß.
Ein Beispiel: Im Feature ABD (Abschnitt 4.6) wird diese Technik angewendet, wenn der Benutzer die abzukürzenden Rufnummern eingibt. Dort wird `PIC_Null` verfeinert um `PIC_Null_ABD_GetIndex` und `PIC_Null_ABD_GetPhoneNumber`.

Wir haben die zweite Methode für jene Features vorbehalten, die nicht nur einen Hauptzustand des Basissystems verfeinern, sondern sich zum Basissystem nebenläufig verhalten und dabei einen featureinternen Zustand merken müssen. Besonders gut kann man dies im Feature CCBS (vgl. Abschnitt 4.7) erkennen, das zwei Subzustandsvariablen verwendet (`CCBS_State_Sender` und `CCBS_State_Receiver`).

5.1.2 Nebenläufigkeit von Features

In Abschnitt 2.3.2 haben wir zwei Möglichkeiten vorgestellt, wie Features an Detection Points ansetzen können: durch Ersetzen des Übergangs vom DP zum PIC oder durch Anfügen der Erweiterung am DP mit anschließender Rückkehr dorthin. Beide Methoden fanden in unseren Erweiterungen Verwendung. Insbesondere ist allerdings

die zweite interessant, da unter den in dieser Arbeit spezifizierten Features tatsächlich der Fall auftritt, daß nicht nur ein Feature an einem DP ansetzt:
An `DP_0_Answer` setzen sowohl ACB als auch CCBS ein, um Hilfsvariablen zurückzusetzen. Dieses Zurücksetzen der Variablen ist möglich, ohne zum folgenden PIC zu springen. Deshalb kehren beide Features wieder an den DP zurück. Also ist es möglich, daß beide Features nacheinander ihre nötigen Aktionen ausführen. Eine störende Interaktion wird vermieden.
Beide Features müssen sich merken, ob sie diese Transition schon einmal ausgeführt haben, um eine Endlosschleife zu vermeiden. ACB tut dies durch eine Subzustandsvariable (`ACB_Activation`), die im Ablauf des Features vorher auf TRUE gesetzt wird, falls die Hilfsvariable zurückgesetzt werden muß. In der Transition wird die Subzustandsvariable dann auf FALSE gesetzt. Im Prinzip enthält also ACB auch einen Subautomaten, der allerdings nur die Zustände TRUE und FALSE annehmen kann. In CCBS ist die Überprüfung nicht so einfach möglich; hier wird in der Provided-Klausel direkt getestet, ob die Hilfsvariable bereits den geänderten Wert enthält oder nicht.
Die Spezifikation der Features mit Hilfe von Subautomaten war also eine große Hilfe bei der Vermeidung von unerwünschten Interaktionen.

5.1.3 Nicht verdeckbare Transitionen

Die im Basissystem vorhandenen Delay-Transitionen führten nicht nur bei der Untersuchung durch `Decfine` zu Problemen (siehe Kapitel 4.1.1), sondern auch bei den Erweiterungen. Es ist in Estelle nicht möglich, Transitionen so zu spezifizieren, daß sie Delay-Transitionen, die im gleichen Zustand feuerbar sind und die diesen Zustand nicht verlassen, komplett verdecken. Ein Beispiel: In der Originalspezifikation des Basissystems aus [Ill95] wurden die Klingeltöne durch Nachrichten erzeugt, die regelmäßig durch eine Delay-Transition verschickt wurden. Im ersten Versuch der Spezifikation des PN-Features wurde eine analoge Delay-Transition eingeführt, die einen anderen Klingelton regelmäßig erzeugt. Diese hatte zwar eine höhere Priorität, konnte die Originaltransition aber nicht verdecken, so daß *zwei* unterschiedliche Klingeltöne an das CCAF verschickt wurden.
Ein Verdecken wäre theoretisch durch eine Transition mit höherer Priorität möglich, die ständig feuern kann, aber weder Zustand noch Variablen ändert, bis die gewünschte Transition aus dem neuen Feature feuerbereit ist oder der Zustand verlassen wird. Aufgrund der höheren Priorität würde immer nur diese Transition zum Schalten angeboten, die Originaltransition käme nie zum Zuge. Dieses Verfahren verbietet sich aber, wenn die Spezifikation mit den üblichen, automatischen Methoden implementiert wird, denn diese Transition würde das System sehr ineffizient machen.
Also durfte der Zustand, in dem diese Delay-Transition feuerbar ist, gar nicht erst erreicht werden. Das hat zur Folge, daß dieser Zustand komplett bis auf diese Transition kopiert werden muß und alle Transitionen, die ihn anspringen, überdeckt werden müssen. Im kopierten Zustand können dann die Erweiterungen angefügt werden. Unerwünschte Interaktionen sind also vorprogrammiert, weil andere Features den

gewünschten Zustand nicht mehr vorfinden. In unserem konkreten Fall löste sich das Problem dadurch, daß die regelmäßig schaltenden Delay-Transitionen ohnehin entfernt wurden (siehe Abschnitt 4.1.1).
In PIC 3 (`Analyze_Information`) tauchte das gleiche Problem mit spontanen Transitionen auf. Hier sollte der Zustand verlassen werden, wenn die Telefonnummer komplett gewählt wurde. Festgestellt wird dies durch eine Pascal-Funktion, die in der Provided-Klausel der Transition, die aus PIC 3 hinausführt, abgefragt wird. Im Basissystem bestehen die Nummern nur aus einer Ziffer, mit dem Feature LongNumber aber aus acht. Ohne ein Kopieren des Zustands für das Feature feuerte die Transition des Basissystems immer nach der ersten gewählten Ziffer. Der Benutzer hatte nie die Möglichkeit, die weiteren Ziffern einzugeben. Deshalb mußte dieser PIC kopiert werden.
Um dieses Problem zu erleichtern, könnten folgende Ansätze nützlich sein:

1. Die Delay-Transitionen könnten deaktiviert werden, indem andere Transitionen deren Timer stoppen, z. B. indem für jeden Timer einer Delay-Transition eine Boolean-Variable bereitgestellt wird. Zusammen mit der höheren Priorität der neuen Transitionen erzielte dies die gleiche Wirkung wie ein Verdecken der Transitionen.

2. Eine etwas drastischere Lösung wäre es, wenn Transitionen andere Transitionen komplett abschalten könnten.

In beiden Fällen wäre zu überprüfen, ob das Verfahren nicht dem Spezifikationsstil widerspricht, demzufolge aus der ursprünglichen Spezifikation nichts ohne vorherige Analyse gelöscht werden darf. Prinzipiell sollte man aber versuchen, Delay-Transitionen so weit wie möglich zu vermeiden.

5.1.4 Nachrichten und deren Parameter

In den Stilregeln für die Spezifikation haben wir festgelegt, daß das Basissystem zunächst nicht verändert, sondern nur ergänzt werden darf. Das betrifft auch die Nachrichten, die zwischen den Modulen verschickt werden. Es gab mehrere Features, die den bereits vorhandenen Nachrichten wie `Setup` oder `Alerting` neue Parameter hinzufügen wollten. Dies ist nach unserer Einschränkung nicht möglich, da nicht nur die Nachrichten selbst, sondern auch die Transitionen, die die Nachrichten empfangen und verschicken, geändert werden müßten. Also müssen die Originaltransitionen von neuen Transitionen verdeckt werden, die die neuen Nachrichten mit den zusätzlichen Parametern verschicken und empfangen.
Sollten mehrere Features dieselbe Nachricht erweitern wollen, sind Konflikte bereits vorprogrammiert, falls die Features unabhängig voneinander entwickelt wurden, da schließlich immer nur ein Feature die Nachricht abändern kann. Diese Feature-Interaktion tritt völlig unnötigerweise auch dann auf, wenn die Features die Nachricht um den gleichen Parameter erweitern wollen.

In [ITU93a] ist eine Liste von Parametern für die Standardnachrichten, die zwischen den OTBCSMs verschickt werden, aufgestellt. Man könnte diese Parameter zwar a priori alle implementieren und dadurch bereits einige Fälle von Interaktionen vermeiden, weil diese Standardparameter von den betreffenden Features nicht mehr hinzugefügt werden müssen. Das Problem ist damit aber nicht vollständig gelöst, da man nie weiß, ob man tatsächlich alle benötigten Werte mitschickt oder ein Feature eigene Variablen definiert, die es mitschicken will.
Vermutlich könnte ein objektorientierter Ansatz als Grundlage für Estelle anstatt Pascal (oder eine objektorientierte Erweiterung) eine Erleichterung für dieses Problem bringen.

5.1.5 Eine Methode für das Hinzufügen neuer Rufnummern

Bei verschiedenen Gelegenheiten muß überprüft werden, ob eine vom Benutzer eingegebene Rufnummer korrekt ist oder nicht, d. h. ob sie im Numbering-Plan vorgesehen ist. Das Basissystem stellt dazu Funktionen zur Verfügung (`Routing_possible` bzw. `Routing_possible_LongNumber` auf der Anruferseite, `IsValidCalleeNumber` auf der Seite des Angerufenen), die mit der fraglichen Rufnummer als Argument aufgerufen wird und einen entsprechenden Boolean-Wert zurückliefert. Im Automaten gibt es dann zwei Transitionen, die dies überprüfen (`PROVIDED Routing_possible` und `PROVIDED NOT Routing_possible`) und entsprechende Maßnahmen ergreifen. Mehrere Features ändern diesen Numbering-Plan, indem sie neue gültige Rufnummern hinzufügen (z. B. Personal Numbering oder Service 0130 aus [Jer97]). Dabei muß man nicht versuchen, die alten Transitionen zu überdecken und sie und die Funktion für das Feature komplett neu schreiben. Folgendes Verfahren hat sich als besonders einfach und zweckdienlich herausgestellt: Im Feature wird eine neue Funktion definiert, die testet, ob die übergebene Rufnummer eine der durch das Feature eingeführten neuen Nummern ist oder nicht. Dazu kommt eine Transition, die nur im positiven Fall feuert und entsprechend weiter verfährt. Wegen der höheren Priorität der Featuretransitionen gegenüber denen des Basissystems wird dieser Test auch immer vorher ausgeführt. Fällt der Test negativ aus, kann mit den beiden Transitionen des Basisystems normal weitergetestet werden. Dies ist nicht nur einfacher als das Verfahren, komplett zu kopieren: man kann auch nach demselben Prinzip weitere Features neue Nummern hinzufügen lassen, ohne daß es zu unerwünschten Interaktionen kommt. Abbildung 5.1 zeigt ein Beispiel dafür.
Generell stellt sich die Frage, ob diese Verhaltensbeschreibung in allen Details auf dieser Ebene wirklich notwendig ist. Ein transparentes Verhalten bezüglich des Numbering-Plan wäre wohl nützlicher. Auch eine Kapselung der Daten in der Service Data Function (SDF), die dienstbezogene und Netzwerkdaten verwaltet, wäre denkbar.

```
#ifdef LongNumber
  TRANS
    PRIORITY LongNumber_Priority
    FROM PIC_Routing_And_Alerting_SS_Routing_Not_Yet
    TO PIC_Routing_And_Alerting_SS_Routing_Ok
    PROVIDED Routing_possible_LongNumber(DestinationPhoneNumber)
   NAME PhoneNumber_is_OK:
    BEGIN
      {...}
    END;
#endif LongNumber

#ifdef LongNumber
  TRANS
    PRIORITY LongNumber_Priority
    FROM PIC_Routing_And_Alerting_SS_Routing_Not_Yet
    TO DP_Route_Select_Failure
    PROVIDED NOT Routing_possible_LongNumber(DestinationPhoneNumber)
   NAME PhoneNumber_is_not_OK:
    BEGIN
    END;
#endif LongNumber

#ifdef LongNumber
#ifdef PN
  TRANS
    PRIORITY PN_Priority
    FROM PIC_Routing_And_Alerting_SS_Routing_Not_Yet
    TO PIC_Routing_And_Alerting_SS_Routing_Ok
    PROVIDED Routing_possible_PN(DestinationPhoneNumber)
   NAME New_PhoneNumber_of_Feature_PN_found:
    BEGIN
      {...}
    END;
#endif PN
#endif LongNumber

#ifdef LongNumber
#ifdef charging
#ifdef S0130
  TRANS
    PRIORITY S0130_low_Priority
    FROM PIC_Routing_And_Alerting_SS_Routing_Not_Yet
    TO PIC_Routing_And_Alerting_SS_Routing_Ok
    PROVIDED Routing_possible_S0130(DestinationPhoneNumber)
   NAME New_PhoneNumber_of_Feature_S0130_found:
    BEGIN
      {...}
    END;
#endif S0130
#endif charging
#endif LongNumber
```

Abbildung 5.1: Beispiel zu Abschnitt 5.1.5: Die beiden ersten Transitionen überprüfen die „normalen“ Rufnummern auf Gültigkeit oder Ungültigkeit. Die beiden unteren Transitionen fügen für die Features PN und S0130 jeweils eine neue Rufnummer hinzu.

5.2 Anmerkungen zum DP-Prinzip

Detection Points wurden in das Call Model eingeführt, um später hinzugefügten Erweiterungen anzuzeigen, daß bestimmte Punkte im Ablauf des Call Models erreicht wurden, und um Einstiegspunkte für die Features zu bieten [DuVi92]. Dieses Prinzip hat sich bei den von uns spezifizierten Features nur teilweise bewährt. Die Idee, fest definierte Ansatzpunkte für die Features zu bieten, trägt sehr viel zur Strukturierung des Systems bei. Der Teufel steckt jedoch, wie immer, im Detail. Folgende Kritikpunkte sind uns aufgefallen:

- Viele Features werden nicht automatisch dadurch aktiv, daß sie einen bestimmten Zustand des Call Models erreichen. Sie werden vielmehr durch Nachrichten aus anderen Modulen wie des zugehörigen CCAFs oder eines in Verbindung stehenden anderen OTBCSMs aktiviert. Dies passiert in den meisten Fällen nicht in einem DP, sondern in einem PIC. Beispiel: Das Feature ACB, das in PIC 3 aktiv wird, sobald der Benutzer anstatt einer Zifferntaste, die dann normalerweise erwartet wird, die ACB-Taste drückt. Für solche Fälle sind die vorhandenen DPs nutzlos. Es fehlt also noch ein geeigneter DP.

- Manche PICs beinhalten sehr viel (zuviel?) Funktionalität.[1] Bei der Spezifikation werden diese PICs in Subzustände verfeinert. Viele Features müssen nur in einen dieser Subzustände eingreifen und lassen die anderen unberührt. Für diese Features ist dann kein geeigneter DP vorhanden, da von diesen nicht mitten in einen PIC gesprungen wird. Den DP des gesamten PICs trotzdem zu verwenden, würde unnötigen Aufwand bedeuten, da alle Subzustände zwischen DP und zu änderndem Subzustand kopiert werden müßten. Dieses Verfahren verbietet sich außerdem, wenn man einen defensiven Spezifikationsstil erreichen will, denn alle Features, die an den dazwischenliegenden Subzuständen ansetzen wollen, würden dadurch unnötigerweise gestört. Es fehlen also noch weitere DPs in manchen PICs.

- Manche PICs wie `PIC_Null` haben mehrere DPs, abhängig davon, aus welchem Vorzustand der PIC erreicht wird. Für diese PICs wäre ein zusätzlicher DP sinnvoll, der in jedem Fall vor Erreichen des PICs durchlaufen wird, unabhängig davon, welcher der Vorzustand war. In den Übergang von diesem DP in den PIC hätten wir z. B. die Transition `PIC_Null_Reset_Variables` einbauen können (siehe auch Abschnitt 4.1.4), die genau einmal bei Eintritt in `PIC_Null` ausgeführt werden soll. Ohne einen solchen PIC mußten wir einen etwas umständlichen Mechanismus spezifizieren, damit die Transition in `PIC_Null` und nur einmal ausgeführt wird.

[1] Dies fällt insbesondere dann auf, wenn man das von uns benutzte Call Model des CS-1 aus [ITU93d] mit dem Basis-Modell aus [ITU93b] vergleicht.

- Aber auch der umgekehrte Fall kam vor: PIC 9 (`T_Alerting`) hat *keinen* DP. Für diesen PIC hätten wir allerdings einen DP benötigt. Stattdessen mußten wir in allen Transitionen aus PIC 8, mit denen zu PIC 9 gesprungen wird, eingreifen und damit einen PIC verändern, der mit den Features eigentlich nichts zu tun hatte.[2]

Es fehlen also noch wichtige DPs, und die vorhandenen sind zum Teil ungeschickt angeordnet.

[2]Nach der Entfernung der Delay-Transitionen (s. 4.1.1) war das Eingreifen in den Vorgänger-PIC nicht mehr nötig, da wir dann nur noch einen der Substates aus der Verfeinerung von PIC 9 ändern mußten. Womit wir wieder beim zweiten • wären.

Kapitel 6

Zusammenfassung

Diese Projektarbeit beschäftigte sich mit der Erweiterung von Telefonvermittlungssystemen. Ziel dieser Arbeit war es, eine vorhandene Estelle-Spezifikation eines solchen Telefonvermittlungssystems um mehrere neue Dienste (Features) zu erweitern. Gleichzeitig wurde in einer zweiten Arbeit dasselbe System um andere Dienste erweitert. Diese Erweiterungen sollen von einem Werkzeug automatisch untersucht werden, ob zwischen ihnen (unerwünschte) Interaktionen auftreten, d. h., ob sich die einzelnen Erweiterungen gegenseitig stören.

Dazu haben wir in Abschnitt 2.1 das Konzept des Intelligenten Netzwerks vorgestellt. Es wurden das Hauptziel, nämlich die leichte Erweiterbarkeit um neue Dienste, und die Unterteilung des Modells in seine vier Ebenen aufgeführt. Einen besonderen Schwerpunkt haben wir auf die verteilte funktionale Ebene gelegt, da auf ihr bereits ein einfaches Telefonvermittlungssystem in einer früheren Arbeit in Estelle spezifiziert wurde. Diese Arbeit wurde in Abschnitt 2.2 vorgestellt.
In Abschnitt 2.3 haben wir Regeln und Möglichkeiten für den Spezifikationsstil aufgelistet. Die Regeln sagen vor allem aus, daß dem System nur Neues hinzugefügt werden darf, und dies auf grobkörnige Art und Weise. Die Möglichkeiten betreffen Zustandsverfeinerungen und Arten des Ansetzens der Features an Detection Points.

Nach dieser Einführung haben wir in Kapitel 3 Prinzipien für Änderungen am Basissystem vorgestellt. Manchmal ist es wünschenswert, das Basissystem, entgegen der Regel aus Abschnitt 2.3, im nachhinein zu verfeinern und damit zu ändern, weil Features keinen geeigneten Ansatzpunkt finden können. Für diesen Fall haben wir eine Vorgehensweise definiert, nach der solche Transformationen durchzuführen sind, ohne daß es zu Problemen kommt, weil sich evtl. das Verhalten des Systems ändert.

In Kapitel 4 haben wir zunächst einige Korrekturen am bestehenden Basissystem aufgezählt, die in erster Linie nötig waren, um dem Analysewerkzeug seine Aufgabe zu erleichtern. Danach stellten wir die einzelnen Erweiterungen vor. Sie sind: Der Ausbau der Rufnummern auf mehrere Ziffern; ein zusätzlicher Klingelton für

unterschiedliche Gruppen von Anrufern; ein zusätzlicher Klingelton für verschiedene Benutzer desselben Endgerätes; automatischer Anruf an denjenigen, der als letzter versucht hat, den Benutzer zu erreichen; die Abkürzung von Rufnummern; eine Benachrichtigung, wenn ein besetzter Anschluß frei wird. Diese Erweiterungen wurden als neue Features in die vorhandene Estelle-Spezifikation erfolgreich integriert.

Unsere Ergebnisse und Erfahrungen haben wir in Kapitel 5 zusammengefaßt. Für den Spezifikationsstil mit Estelle haben wir folgendes herausgefunden:

- Das Prinzip, Zustandsverfeinerungen mit einer Namenskonvention bei den Estelle-Hauptzuständen durchzuführen, hat sich bewährt.
- Features, die parallel zum Basissystem ablaufen, lassen sich gut durch Subautomaten mit Hilfe von Zustandsvariablen spezifizieren.
- Subzustände sind auch eine Hilfe, um Features an den Detection Point, an dem sie angesetzt hatten, zurückkehren zu lassen. Diese Rückkehr, im Gegensatz zu einem Übergang zum nächsten Hauptzustand, vermeidet unerwünschte Feature-Interaktionen.

An Estelle selbst kritisieren wir zwei Eigenschaften:

- In Estelle ist es nicht immer möglich, Transitionen des Basissystems durch Prioritäten zu verdecken, damit ein Feature etwas ändern oder ergänzen kann. Es fehlt ein Mechanismus, um bestimmte Transitionen, z. B. Delay-Transitionen, am Schalten zu hindern.
- Das Hinzufügen von neuen Parametern in Nachrichten verursacht großen Aufwand. Ein objektorientierter Ansatz könnte Probleme vermeiden.

Außerdem haben wir eine Methode entwickelt, wie man auf einfache Art und Weise dem System neue gültige Rufnummern hinzufügen kann, ohne daß Änderungen am Basissystem vorgenommen werden müssen oder verschiedene neue Nummern aus mehreren Features sich gegenseitig stören.

Zum Prinzip der Detection Points läßt sich festhalten, daß es sich grundsätzlich bewährt hat. Die DPs tragen viel zur Strukturierung des Systems bei. Allerdings fehlen sie noch an einigen Stellen, an anderen sind sie ungeschickt angeordnet.

Zukünftige Arbeiten sollen mit Hilfe dieser Arbeit zeigen, inwiefern es praktisch durchführbar ist, Feature-Interaktionen mit Analysewerkzeugen automatisch finden zu lassen.

Literaturverzeichnis

[Bar96] Barthel, D. *Implementation von Kriterien zur Erkennung und Einstufung von Feature-Interaktionen.* Diplomarbeit, Univ. Kaiserslautern, FB Informatik (Aug. 1996).

[Bre95a] Bredereke, J. *Erweiterung eines EA an DPs.* Interne Notiz (Jan. 1995).

[Bre95b] Bredereke, J. *Verfeinerung von Verhalten in Estelle.* Interne Notiz (Jan. 1995).

[Bre97] Bredereke, J. *Communication Systems Design with Estelle - On Style, Efficiency, and Analysis.* Dissertation, Shaker Verlag, Aachen, ISBN 3-8265-2764-X (Apr. 1997).

[DuVi92] Duran, J. M. und Visser, J. *International Standards for Intelligent Networks.* IEEE Commun. Mag. **30**(2), 34-42 (Feb. 1992).

[Got95] Gotzhein, R. *Spezifikation von Kommunikationssystemen.* Skriptum zur Vorlesung, Univ. Kaiserslautern, FB Informatik (1995).

[Ill95] Illerich, J. *Entwurf eines Telefonvermittlungssystems in Estelle.* Projektarbeit, Univ. Kaiserslautern, FB Informatik (Okt. 1995).

[ITU93a] ITU-T, *Recommendation Q.931, DSS 1 - ISDN User-Network Interface for Basic Call Control.* (März 1993).

[ITU93b] ITU-T, *Intelligent Network Recommendation Q.1204 Intelligent Network Distributed Functional Plane Architecture.* (März 1993).

[ITU93c] ITU-T, *Intelligent Network Recommendation Q.1211 Introduction to Intelligent Network Capability Set 1.* (März 1993)

[ITU93d] ITU-T, *Intelligent Network Recommendation Q.1214 Distributed Functional Plane For Intelligent Network CS-1.* (März 1993).

[ITU93e] ITU-T, *Q12xx-Series Intelligent Network Recommendations.* (1993).

[Jer97] Jerusalem, D. *Erweiterung eines Telefonvermittlungssystems in Estelle.* Projektarbeit, Univ. Kaiserslautern, FB Informatik (Aug. 1997).

[UniKL97] *Personal- und Telefonverzeichnis der Universität Kaiserslautern.* Univ. Kaiserslautern (Mai 1997).