Spezifikation und Bewertung eines effizienten Kommunikationsprotokolls

Stephan Dahl

Februar 1995

Zur schnellen Kommunikation zwischen Rechnern werden laufzeiteffiziente Implementationen von Protokoll-Spezifikationen benötigt. Die herkömmliche Schichten-Aufteilung verursacht hohe Kosten. In dieser Projektarbeit wurde eine andere Spezifikationsform, die Methode des strukturierten Produktautomaten, am Beispiel der OSI-Schichten 5 und 6 untersucht. Der Aufwand zur Erstellung und Wartung der Spezifikation und die Laufzeiteffizienz der daraus entstandenen Inplementation wurden mit mehreren anderen Spezifikationsformen verglichen und bewertet. Die Methode des strukturierten Produktautomaten erwies sich dabei als ein geeigneter Spezifikationsstil.

Fachbereich Informatik
AG Rechnernetze
Prof. R. Gotzhein
Aufgabenstellung und Betreuung: Dipl. Inform. J. Bredereke

Erklärung

Ich versichere hiermit, die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Kaiserslautern, den

Inhaltsverzeichnis

1 Einleitung

Beim Datenaustausch zwischen Rechnern über ein Netz müssen sich beide Seiten an festgelegte Regeln halten, damit ein reibungsloser, fehlerfreier Ablauf stattfinden kann. Diese Regeln sind in sogenannten Protokollen zusammengefaßt. Das Gesamtprotokoll oder auch Kommunikationsmodell für die Kommunikation zwischen Rechnern ist dabei oftmals aus mehreren Schichten aufgebaut, wobei auf jeder Schicht ein eigenes Protokoll abläuft.

Früher waren die Übertragungsmedien des Netzes, z.B. Kabel oder Telefonleitungen, sehr langsam im Vergleich zu den Rechnern. Deshalb war der Protokollaufwand, d.h. die Zeit für die Abwicklung der Protokolle auf dem Rechner, ohne Bedeutung für den Gesamtkommunikationsaufwand. Heute dagegen stehen sehr schnelle Übertragungsmedien zur Verfügung, z.B. Glasfaserkabel. Dadurch bekommt der Protokollaufwand einen bedeutenden Anteil am Gesamtaufwand, und eine möglichst effiziente Implementierung der Protokolle wird nötig.

In dieser Projektarbeit wurde als Kommunikationsmodell das von der ISO (International Organization for Standardization) genormte OSI-Schichtenmodell (Open Systems Interconnection - Schichtenmodell) gewählt. Eine eindeutige Spezifikation erfolgte durch die Beschreibung in der FDT (Formal Description Technique) Estelle. Mit Hilfe von verschiedenen Software-Werkzeugen, z.B. Compiler oder Interpreter, lassen sich aus dieser Spezifikation auch ausführbare Implementierungen erzeugen. Bei einer herkömmlichen Implementierung wird dabei jede Schicht des Kommunikationsmodells als eigenständiger Automat generiert. Ein möglicher Ansatz zur Verminderung der Zeit, die benötigt wird um das Protokoll abzuwickeln, ist das Verschmelzen von mehreren Schichten in einem Automaten. Werden zwei Schichten miteinander verschmolzen, so spricht man auch von der Bildung des Produktautomaten. Diese Vorgehensweise hat aber den Nachteil eines hohen Aufwands bei der Erstellung der Spezifikation.

In der Projektarbeit wurde ein anderer Ansatz verfolgt, Schichten zu verschmelzen, und zwar die Bildung eines strukturierten Produktautomaten. Eine vorhandene herkömmliche Estelle-Spezifikation der Schichten 5 und 6 des OSI-Schichtenmodells wurde so umgearbeitet, daß diese Schichten in einem Automaten integriert werden konnten. Vergleichsmessungen der entstandenen Implementation mit einer herkömmlichen Spezifikation, in der beide Schichten durch jeweils einen eigenen Automaten realisiert wurden, und einer Produktautomat-Spezifikation wurden durchgeführt. Die Meßergebnisse wurden ausgewertet und ins Verhältnis zum Erstellungsaufwand gesetzt. Dabei zeigte sich die Leistungsfähigkeit dieses Ansatzes.

In Kapitel 2 wird das OSI-Schichtenmodell vorgestellt, wobei insbesondere auf die beiden Schichten 5 und 6, die Sitzungs– und die Anwendungs–Schicht, eingegangen wird. Die angebotenen Dienste dieser Schichten werden erläutert, ein vereinfachter Automat der jeweiligen Schicht wird besprochen, sowie der genaue Ablauf eines Verbindungsaufbaus in beiden Schichten wiedergegeben.

Kapitel 3 beschäftigt sich mit der formalen Beschreibungstechnik (FDT) Estelle. An einem Beispiel zweier miteinander kommunizierender Automaten wird gezeigt, wie eine herkömmliche Spezifikation dieses Kommunikationsmodells in Estelle erfolgt.

Die Kapitel 4 und 5 führen an diesem Beispielmodell zwei Möglichkeiten zur effizienteren Implementation vor. Die Bildung eines Produktautomaten wird in Kapitel 4 formal definiert und auf das Beispiel angewandt. Ebenso wird in Kapitel 5 verfahren. Nach der Erläuterung der Idee des strukturierten Produktautomaten, wird das Beispiel in diese Spezifikationsform transformiert.

Kapitel 6 beschreibt notwendige Änderungen, sowie die prinzipielle Vorgehensweise, die es ermöglichen, aus einer herkömmlichen Estelle-Spezifikation der Schichten 5 und 6, einen strukturierten Produktautomaten zu erhalten.

In Kapitel 7 wird die Meßumgebung für die Vergleichsmessungen der unterschiedlichen Spezifikationsformen beschrieben.

Im letzten Kapitel, dem Kapitel 8, wird die Methode des strukturierten Produktautomaten zur effizienteren Implementation von Kommunikationsmodellen anhand der erhaltenen Meßergebnisse mit den anderen Spezifikationsformen verglichen und bewertet.

2 Das OSI-Schichten-Modell

Eine Netzwerk-Architektur, die nach dem ISO-OSI-Schichten-Modell, dem Basis-Referenz-Modell [ISO7498] aufgebaut ist, besitzt sieben Schichten. Der Gesamtaufbau und die Namen der Schichten sind in Tabelle 1 dargestellt.

Nr. 7	Anwendung
6	Darstellung
5	Sitzung
4	Transport
3	Vermittlung
2	Sicherung
1	Bitübertragung

Tabelle 1: Schichten-Modell

Kommunizieren zwei Benutzer mit Hilfe des OSI-Schichten-Modells miteinander, so kommunizieren die einzelnen Partner-Schichten der beiden Benutzer, indem sie die Dienste der direkt darunterliegenden Schicht nutzen. Erst die Bitübertragungsschichten kommunizieren über ein physikalisch vorhandenes Übertragungsmedium, wie z.B. Telefonleitungen, Koaxialkabel oder Glasfaserleiter.

Jede Schicht stellt einen Dienstzugangspunkt (SAP Service access point) zur Verfügung, über den die Dienste der Schicht angefordert werden können und an dem eintreffende Ereignisse, wie z. B. ein Verbindungsaufbauwunsch eines anderen Nutzers, angezeigt werden. In Abbildung 1 ist die Verbindungsstruktur dreier aufeinanderfolgender Schichten gezeigt.

Es darf immer nur die direkt darüberliegende Schicht (Schicht n+1) den Dienstzugriffspunkt benutzen und die Schicht selbst (Schicht n) nur die Dienste der direkt darunterliegenden Schicht (Schicht n-1) nutzen.

Will die Schicht n+1 Dienste der Schicht n-1 nutzen, so ist dies nur möglich, wenn die Schicht n diese Dienste durchreicht. Dieses Durchreichen von Diensten kommt in den oberen Schichten des OSI-Modells häufig vor.

In dieser Projektarbeit wurden nur die Schichten 5 und 6 bearbeitet. In den folgenden Abschnitten werden darum auch nur diese Schichten ausführlicher beschrieben. Die Aufgaben der anderen Schichten, sowie deren Funktionsumfang, sind in [Tan92] bzw. in den einzelnen ISO-Normen zu den Schichten zu finden.

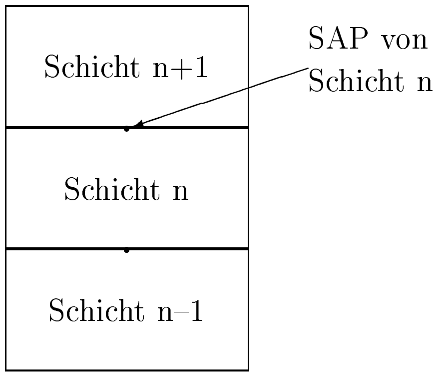

Abbildung 1: Verbindungsstruktur

2.1 Schicht 5

Die Sitzungs-Schicht ist im OSI-Schichten-Modell die Schicht mit der zugeordneten Nummer 5. Ihre Spezifikation erfolgte durch die ISO-Normen [ISO8326] und [ISO8327]. Sie erweitert die von der Transportschicht angebotenen Dienste, den Verbindungsaufbau, die fehlerfreie Datenübertragung sowie den Verbindungsabbau, um folgende Komponenten:
Dialogverwaltung, Synchronisierung, Aktivitätsverwaltung, sowie die Ausnahmeberichterstattung.
Die Dialogverwaltung erlaubt einen Halbduplexbetrieb einer Sitzungs-Schicht Verbindung auf einer im Normalfall vorhandenen Vollduplex-Transport Verbindung. Dies wird oftmals für eine vereinfachte Implementation von Benutzerprozessen benötigt. Die Synchronisierung dient der Fehlerbehebung in höheren Schichten des OSI-Modells, also alle Fehler, die nichts mit verlorengegangenen oder beschädigten Nachrichtenpaketen zu tun haben, da diese Art von Fehlern ab der Transport-Schicht nicht mehr vorkommen können. Die Aktivitätsverwaltung erlaubt die Einteilung des Nachrichtenstroms in voneinander unabhängige logische Einheiten, z.B. bei einem Programm das Dateien von einem Rechner zu einem anderen überträgt ist jede Datei eine logische Einheit. Wird die Übertragung bei irgendeiner Datei gestört, z.B. durch Rechnerausfall, so brauchen die bereits komplett übertragenen Dateien nicht nochmal beim nächsten Versuch gesendet werden.

In jeder Schicht gibt es sogenannte Kerndienste, d.h. Dienste, die in jeder Implementierung vorhanden sein müssen. Alle anderen Dienste der Schicht sind optional und brauchen in einer korrekten Spezifikation der Schicht nicht definiert werden. Pro Dienst können vier verschiedene Dienstelemente definiert sein, die verschiedene Aktionen dieses Dienstes repräsentieren. Es sind dies die Anforderung (req), die Anzeige (ind), das Antworten (rsp) und die Bestätigung (cnf) des jeweiligen Dienstes. Dabei werden die Anforderung und das Antworten vom Nutzer, hier also Schicht 6, der Dienste initiiert, während es bei der Anzeige und der Bestätigung durch den Anbieter, hier also die Schicht 5, geschieht.

In der Tablelle 2 sind die Kerndienste von Schicht 5 aufgezählt.

Dienst	Bezeichnung	Dienstelemente
Verbindungsaufbau	SCON	req, ind, rsp, cnf
Datenübertragung	SDT	req, ind
geordneter Abbau	SREL	req, ind, rsp, cnf
abrupter Abbau durch Nutzer	SUAB	req, ind
abrupter Abbau durch Anbieter	SPAB	ind

Tabelle 2: Kerndienste der Sitzungs-Schicht

Im Rahmen der Projektarbeit wurden nur die Dienste des Verbindungs-Aufbaus, des -Abbaus sowie der Datenübertragung, also die Kernfunktionen der Schicht, implementiert. Deshalb werden auch nur diese hier näher betrachtet.

Der Verbindungsaufbau und die Datenübertragung wurden in ihrer Funktionalität direkt von der Transport-Schicht übernommen. Der Verbindungsabbau dagegen wurde verändert. Die Transport-Schicht bietet nur den abrupten Verbindungsabbau an, d.h. hat ein Partner die Verbindung abgebaut, werden keine Daten mehr empfangen und der Gegenüber hat keine Möglichkeit den Abbauwunsch abzulehnen. Ab der Sitzungs-Schicht steht der geordnete Abbau zur Verfügung, der alle diese Nachteile beseitigt. Ein Datenverlust durch einen Verbindungsabbauwunsch ist nicht mehr möglich, da bis zum Eintreffen einer positiven Bestätigung durch den Partner noch Datenpakete empfangen werden. Auch hat der Gegenüber die Möglichkeit mit einer negativen Bestätigung die Verbindung weiter bestehen zu lassen. Abbildung 2 zeigt die Sitzungs-Schicht und ihre Nachbarschichten sowie die Dienstzugangspunkte.

Die Dienstelemente, die zur Realisierung der Kerndienste der Sitzungs-Schicht zwischen den Schichten an den Schnittstellen (1) und (2) in Abbildung 2 ausgetauscht werden, sind in Tabelle 3 aufgelistet.

Dabei bedeutet SPDU Session Protocol Data Unit. Diese werden in Datenpakete verpackt und durch den Datenübertragungsdienst der Transport-Schicht verschickt. Sie dienen ausschließlich zur Kommunikation zwischen zwei Sitzungs-Schichten, sie werden also innerhalb einer Sitzungs-Schicht erzeugt und von einer anderen verarbeitet. Mittels dieser Kommunikationsmöglichkeit können die Sitzungs-Schichten z.B. beim Verbindungsaufbau über bestimmte Parameter der Verbindung verhandeln oder einen geordneten Verbindungsabbau realisieren.

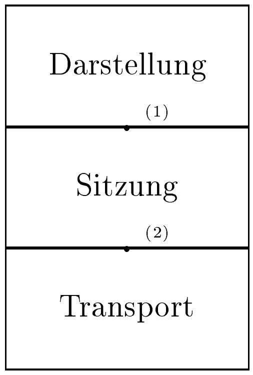

Abbildung 2: Sitzungs-Schicht und SAPs

In der Tabelle 3 sind nicht alle Dienstelemente der Kerndienste der Schicht 5 aufgeführt, sondern nur diejenigen, die im Transitionsdiagramm aus Abbildung 3 enthalten sind. Es fehlen die Dienstelemente des abrupten Verbindungsabbaus, sowie die negativen Antwort- und Bestätigungs-Dienstelemente. Diese wurden weggelassen um das Transitionsdiagramm für den Automaten der Schicht 5 übersichtlicher zu gestalten. In Abbildung 3 ist also der Automat der Kerndienste der Sitzungs-Schicht abgebildet, unter der Einschränkung, daß nur der positiv beantwortete Verbindungs-Aufbau und geordnete -Abbau, sowie die Datenübertragung berücksichtigt wurden.

Der Automat der vollständigen Spezifikation der Kerndienste der Sitzungs-Schicht besitzt gegenüber dem Automaten aus Abbildung 3 noch zusätzliche Zustände und Transitionen um die kompletten Kerndienste abarbeiten zu können. Einzelne Kerndienste werden in diesem Automaten etwas anders realisiert, z.B. ist es beim Verbindungsaufbau möglich, mehr Nutzerdaten zu übertragen als in eine SPDU CN hineinpassen. Dies geschieht mit Hilfe der SPDUs OA und CDO. In der Spezifikation der Schicht durch die ISO wurde optional vorgesehen beim Verbindungsabbau die bestehende Transportverbindung nicht abzubauen und gegebenenfalls bei einem erneuten Verbindungsaufbau zum gleichen Kommunikationspartner wiederzuverwenden.

Die Zuordnung der Zustandsbezeichnungen aus Abbildung 3 und den Bezeichnungen aus der ISO-Norm [ISO8327] kann man der Tabelle 4 entnehmen.

Die Transitionsdiagramme der Schichten 5 und 6 wurden aus einer Spezifikation dieser beiden Schichten in Estelle abgeleitet. Bei der Sitzungs-Schicht ergibt sich ein Unterschied zwischen Estelle-Spezifikation und ISO-Norm, weshalb für den Automatenzustand A_TDISind keine äquivalente Bezeichnung aus der Norm gefunden wurde. Es stand allerdings nur die erste Fassung der ISO-Normen zur Verfügung. Inwieweit sich eine bereits existierende zweite Fassung

Schnittstelle	Dienstelement	Kommentar
(1) zwischen Darstellungs- und Sitzungs- Schicht	SCONreq SCONind SCONrsp+ SCONcnf+	Verbindungs- Aufbau
	SDTreq SDTind	Datenüber- tragung
	SRELreq SRELind SRELcnf+ SRELrsp+	geordneter Verbindungs- Abbau
(2) zwischen Sitzungs- und Transport- Schicht	TCONreq TCONind TCONrsp+ TCONcnf TDISreq TDISind	Dienst- elemente der Transport- Schicht
	TDT(CN) TDT(AC) TDI(DT) TDT(FN) TDT(DN)	In TDT eingepackte SPDUs der Sitzungs- Schicht

Tabelle 3: Dienstelemente-Zuordnung

und die Estelle-Spezifikation unterscheiden konnte leider nicht festgestellt werden.

Abbildung 4 zeigt den genauen Abblauf eines Verbindungsaufbaues zwischen einem Nutzer A und einem Nutzer B durch die Sitzungs-Schicht. Die Zeit nimmt von oben nach unten zu.

Nutzer A ist der Initiator, d.h. er fordert seine Sitzungs-Schicht auf, eine Verbindung zu Nutzer B herzustellen (SCONreq). Die Sitzungsschicht fordert daraufhin von der darunterliegenden Transport-Schicht eine Verbindung an (TCONreq). Der Sitzungs-Schicht des Nutzers B wird der Verbindungsaufbauwunsch von A angezeigt (TCONind) und diese bestätigt den Aufbauwunsch einer Transportverbindung positiv (TCONrsp+). Nachdem diese Antwort die Sitzungs-Schicht von A erreicht hat (TCONcnf+), schickt diese die SPDU CN (Connection) an die Sitzungs-Schicht von B. In dieser SPDU sind die Parameter der Verbindung enthalten. Diese werden dem Nutzer B gemeldet (SCONind) und seine Antwort Nutzer A übermittelt.

Falls Nutzer B mit dem Verbindungsaufbauwunsch, sowie den Parametern der Verbindung einverstanden ist (SCONrsp+), schickt seine Sitzungs-Schicht der

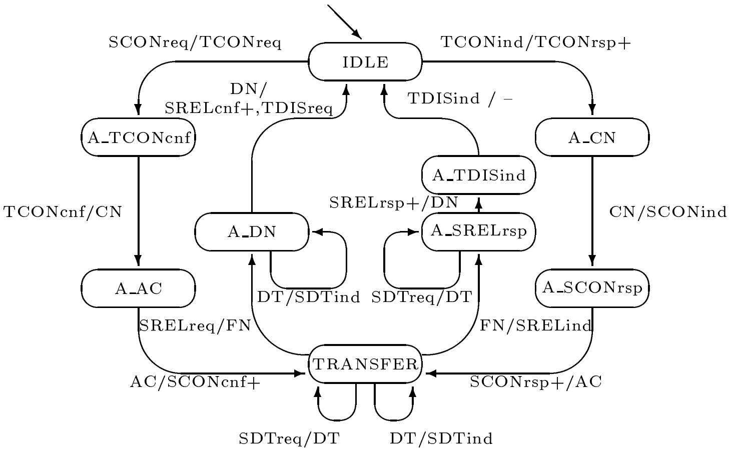

Abbildung 3: Automat der Sitzungs-Schicht

Schicht von A die SPDU AC. Die Sitzungs-Schicht von A informiert dann den Nutzer A von dem erfolgreichen Verbindungsaufbau zu B (SCONcnf+).

2.2 Schicht 6

Die Schicht mit der Nummer 6 des OSI-Schichten-Modells ist die Darstellungs-Schicht. Sie wurde in den ISO-Normen [ISO8322] und [ISO8323] spezifiziert. Ihre Hauptaufgabe besteht in der Konvertierung der zu übermittelnden Daten in die jeweiligen rechnerabhängigen Darstellungsformate, ohne die Bedeutung

Zustandsbezeichnung	ISO–Bezeichnung
IDLE	STA01
A_TCONcnf	STA01B
A_AC	STA02A
TRANSFER	STA713
A_CN	STAI01C
A_SCONrsp	STAI08
A_DN	STA03
A_SRELrsp	STA09
A_TDISind	?

Tabelle 4: Zustandsbezeichnungen

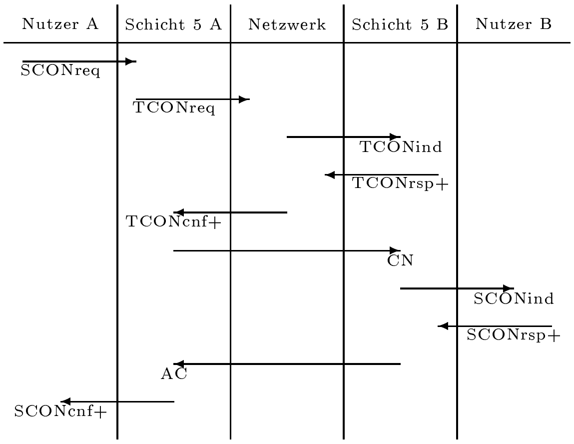

Abbildung 4: Verbindungsaufbau in Schicht 5

der Daten zu ändern. Zu diesem Zweck wurde eine rechnerunabhängige Beschreibung von Datenstrukturen entwickelt und standardisiert, die abstrakte Syntaxnotation 1, kurz ASN.1. Neben dem reinen Konvertieren von Daten zählen auch noch die Verschlüsselung und Komprimierung der Daten, sowie das Durchreichen aller von der darunter liegenden Schicht, der Sitzungsschicht, angebotenen Dienste, zu ihrem Funktionsumfang.

Die Schicht 6 besitzt die in der Tabelle 5 aufgelisteten Kerndienste.

Dienst	Bezeichnung	Dienstelemente
Verbindungsaufbau	PCON	req, ind, rsp, cnf
Datenübertragung	PDT	req, ind
geordneter Abbau	PREL	req, ind, rsp, cnf
abrupter Abbau durch Nutzer	PUAB	req, ind
abrupter Abbau durch Anbieter	PPAB	ind

Tabelle 5: Kerndienste der Darstellungsschicht

Auch hier werden nur die Dienstprimitive näher beschrieben, die im Rahmen der Projektarbeit implementiert wurden. Dies waren nur die Kernfunktionen, also die Dienstprimitive, die eine Verbindung aufbauen, Daten übertragen und die Verbindung abbauen. Alle sonstigen Dienste der Darstellungsschicht, einschließlich der impliziten Konvertierung der Daten in ASN.1-Notation bei der Datenübertragung, werden nicht berücksichtigt.

Die Dienstelemente, die zur Realisierung der Kerndienste der Darstellungs-Schicht, zwischen den Schichten an den Schnittstellen (1) und (2) in Abbildung 5 ausgetauscht werden, sind in Tabelle 6 aufgelistet.

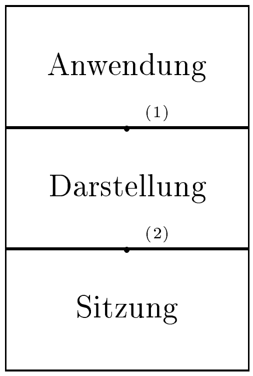

Abbildung 5: Darstellungs-Schicht und SAPs

Der abrupte Verbindungsabbau, sowie negative Bestätigungen von Dienstanforderungen sind in der Tabelle 6 nicht berücksichtigt. Auf diese wurde verzichtet, um den Automaten der Darstellungsschicht, welcher in Abbildung 6 dargestellt ist, übersichtlicher zu gestalten.

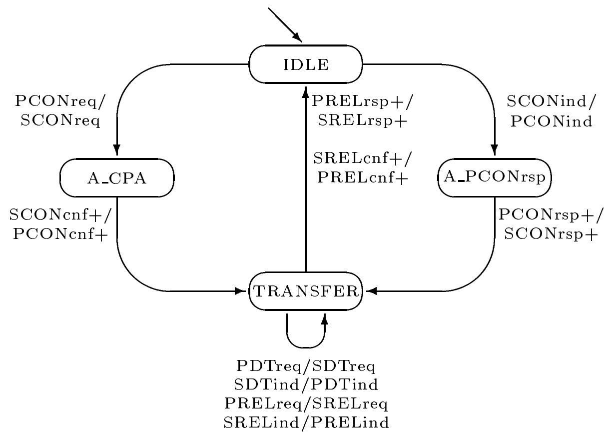

Abbildung 6: Automat der Darstellungsschicht

Der Zustand IDLE ist der Anfangszustand, in dem der Automat keine Verbindung zu einem anderen hat und auch noch keine Dienstanforderung vorliegt.

Wird der Automat von der darüberliegenden Schicht, d. h. dem Benutzer, aufgefordert eine Verbindung aufzubauen (PCONreq), so wird die linke Transitionsfolge über den Zustand A_CPA zum Zustand TRANSFER ausgeführt.

Kommt dagegen eine Verbindungsanforderung von der Sitzungsschicht

Schnittstelle	Dienstelement	Kommentar
(1) zwischen Anwendungs- und Darstellungs-Schicht	PCONreq PCONind PCONrsp+ PCONcnf+	Verbindungs-Aufbau
	PDTreq PDTind	Datenübertragung
	PRELreq PRELind PRELcnf+ PRELrsp+	geordneter Verbindungs-Abbau
(2) zwischen Darstellungs- und Sitzungs-Schicht	SCONreq SCONind SCONrsp+ SCONcnf+ SDTreq SDTind SRELreq SRELind SRELcnf+ SRELrsp+	Dienst-elemente der Sitzungs-Schicht

Tabelle 6: Dienstelemente-Zuordnung

(SCONind), d. h. von einem anderen Nutzer, wird die rechte Transitionenfolge über A_PCONrsp zu TRANSFER durchlaufen.

Im Zustand TRANSFER können dann beide Automaten Daten übertragen, bis einer der beiden von seinem Nutzer den Auftrag erhalten hat die Verbindung zu beenden und der andere dies bestätigt. Danach befinden sich beide Automaten wieder im Anfangszustand IDLE.

Es wurden nur Transitionen mit positiven Bestätigungen (–rsp+ oder –cnf+) von Dienstanforderungen in den Graphen aufgenommen, um ihn übersichtlicher zu halten. Falls beim Verbindungsaufbau eine Anforderung negativ beantwortet wird, gehen beide Automaten in den Anfangszustand IDLE zurück. Bei einer Ablehnung des Verbindungsabbaus verbleiben beide Automaten im Zustand TRANSFER und können somit auch weiterhin Daten austauschen.

Der vollständige Automat, der auch negative Bestätigungen verarbeitet, sowie den abrupten Verbindungsabbau enthält, hat gegenüber diesem Automaten keine zusätzlichen Zustände, es sind nur zusätzliche Transitionen notwendig um die noch fehlenden Dienstelemente zu realisieren. In der Tabelle 7 sind die Zuordnungen der Zustandsbezeichnungen des Automaten aus Bild 6 zu denen aus der ISO-Norm [ISO8323] aufgelistet.

Zustandsbezeichnung	ISO–Bezeichnung
IDLE	STAI0
TRANSFER	STAt0
A_CPA	STAI1
A_PCONrsp	STAI2

Tabelle 7: Zustandsbezeichnungen

Zur besseren Veranschaulichung, wie ein Verbindungsaufbau zwischen einem Nutzer A und einem Nutzer B stattfindet, ist in Bild 7 der Ablauf in einem Zeitdiagramm dargestellt. Darin sind alle Dienstprimitive, die zwischen Nutzern A und B und ihren jeweiligen Darstellungs-Schichten Schicht 6A und Schicht 6B, sowie zwischen den beiden Darstellungs-Schichten der Nutzer und den darunterliegenden, hier als Netzwerk bezeichneten Schichten, ausgetauscht werden, enthalten. Die Zeit schreitet von oben nach unten fort.

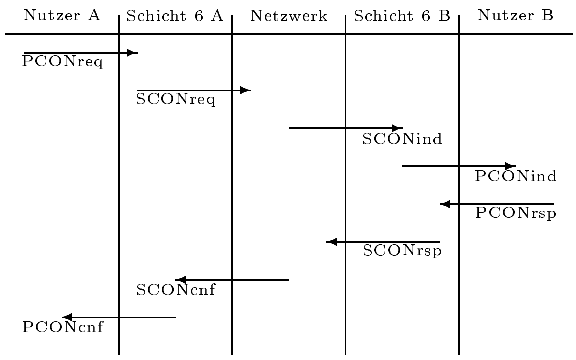

Abbildung 7: Verbindungsaufbau der Darstellungsschicht

Sobald der Nutzer A seine Darstellungsschicht durch das Dienstprimitiv PCONreq auffordert, einen Verbindungsaufbau zu initiieren, fordert diese die darunterliegende Schicht 5 auf, eine Verbindung herzustellen (SCONreq). Diese Aufforderung geht dann über das Netzwerk an die Darstellungsschicht des Nutzers B (SCONind). Diese informiert den Nutzer B über das Eintreffen einer solchen Dienstanforderung (PCONind). Die Antwort des Nutzers B (PCONrsp) wird mittels Schicht 5 (SCONrsp) an die Darstellungsschicht des Nutzers A weitergeleitet (SCONcnf). Der Nutzer A wird über die Antwort von B informiert (PCONcnf). Hat B den Verbindungsaufbauwunsch positiv bestätigt, so besteht nun eine Verbindung zwischen A und B, ansonsten ist der Verbindungsaufbau gescheitert und A kann entsprechend darauf reagieren, z.B. es noch einmal versuchen.

Wie man sieht, erfolgt beim Verbindungsaufbau eine 1-zu-1 Abbildung der Dienstprimitive der Darstellungsschicht auf die der Sitzungsschicht. Auch bei der Datenübertragung und dem Verbindungsabbau ist dies der Fall, d.h. die Darstellungsschicht reicht diese Dienste nur nach oben durch, ohne sie in ihrer Funktionalität zu erweitern.

3 Estelle

Die Beschreibung eines Protokolls durch ein informelles Automatendiagramm oder durch informelle Tabellen ist nicht eindeutig. Die Semantik der Transitionen ist nicht festgelegt, und so sind verschiedene Interpretationen des Protokolls möglich. Eine Beschreibung durch ein Automatendiagramm ist zusätzlich auch noch aufwendig, da die Kontextvariablen darin dargestellt werden müssen.

Eine eindeutige Spezifikation eines Protokolls erhält man durch eine Beschreibung in einer formalen Beschreibungssprache (FDT = formal description technique), in der die Semantik jedes einzelnen Ausdrucks genau festgelegt ist. Eine von der ISO genormte FDT ist Estelle, in der z.B. bereits einzelne Schichten des OSI-Schichtenmodells spezifiziert wurden.

An der Spezifikation eines Beispielautomaten-Paares sollen die wichtigsten Estelle-Konstrukte kurz erklärt werden. Diese Spezifikation ist weder vollständig, noch sind alle Parameter für einzelne Konstrukte richtig gesetzt. Sie dient nur als grobes Schema für den grundsätzlichen Aufbau einer Spezifikation von miteinander kommunizierender Automaten. Das selbe Aufbauschema findet sich bei allen in dieser Projektarbeit enthaltenen Spezifikationen. Eine ausführliche Einführung in die FDT Estelle findet sich in [BuDe87] oder [Got94] oder [Tur93] und die genaue Definition der Syntax sowie der Semantik der einzelnen Konstrukte steht in [ISO9074].

3.1 Beispiel

In diesem Beispiel soll die Spezifikation von zwei miteinander kommunizierender Automaten A und B vorgenommen werden. Abbildung 8 zeigt die Architektur der beiden Automaten. A kommuniziert dabei außer mit B auch noch mit einem nicht näher spezifizierten anderen Automaten. Das gleiche gilt für Automat B.

In den Abbildungen 9 und 10 sind das Verhalten, d.h. das Transitionsdiagramm, der Automaten A und B abgebildet.

Jede Kommunikationsverbindung der Automaten wird in der Estelle-Spezifikation auf einen Kanal abgebildet. Durch jeden Kanal können nur bestimmte Datenpakete, die sogenannten Interaktionen, von einem Automaten zum anderen gesendet werden. Dies muß für jeden Kanal explizit angegeben werden, und durch die Definition von Rollen wird zusätzlich noch die Richtung, in der diese Interaktionen ausgetauscht werden können, festgelegt. Die entsprechenden Definitionen der Kanäle für die Verbindungen der beiden Automaten A und B sehen folgendermaßen aus:

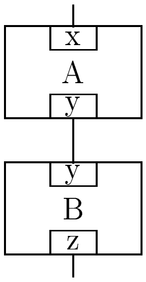

Abbildung 8: Architektur des Beispiels

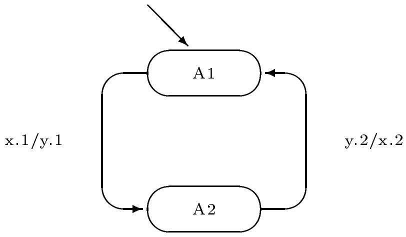

Abbildung 9: Transitionsdiagramm von A

```
channel xKanal(Benutzer, Anbieter);
by Benutzer : x1;
by Anbieter : x2;

channel yKanal(Benutzer, Anbieter);
by Benutzer : y1;
by Anbieter : y2;

channel zKanal(Benutzer, Anbieter);
by Benutzer : z1;
              z3;
by Anbieter : z0;
              z5;
              z4;
```

Das Verhalten eines Automaten wird in Estelle in einem Modul spezifiziert. Jedes Modul besteht aus einem Modulkopf, in dem die Verbindungen zur Umgebung festgelegt werden, und dem Modulrumpf, der das konkrete Verhalten des Automaten beschreibt.

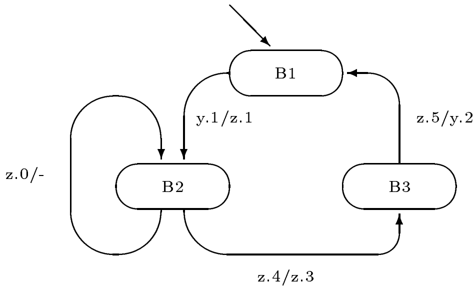

Abbildung 10: Transitionsdiagramm von B

Im Modulkopf gibt man also die Kanäle an, über die der Automat kommuniziert, sowie die Rolle, die er dabei inne hat, d.h. dadurch ist eindeutig definiert, welche Interaktionen er empfangen und welche er senden kann. Für das Beispiel mit den Automaten A und B ergeben sich folgende Modulköpfe:

```
module A_Automaten_Typ;
    ip X : xKanal(Anbieter);
       Y : yKanal(Benutzer);
end;

module B_Automaten_Typ;
    ip Y : yKanal(Anbieter);
       Z : zKanal(Benutzer);
end;
```

Der einem Modulkopf zugeordnete Modulrumpf spezifiziert das Verhalten des Moduls, indem genau festgelegt wird, wie der Automat auf eintreffende Ereignisse, wie z.B. das Empfangen einer Interaktion oder das Starten des Automaten, reagiert. Dazu werden alle Zustände des Automaten aufgelistet, es wird festgelegt, welches der Startzustand ist und welche Aktionen beim Start des Automaten ausgeführt werden. Ferner wird für jeden Zustandsübergang des Automaten eine sogenannte Transitionsdefinition angegeben, die den Ausgangszustand, die den Übergang auslösende Interaktion, den Zielzustand sowie die beim Zustandsübergang auszuführenden Aktionen festlegt.

Den Automaten A beschreibt folgender Modulrumpf:

```
body A_Rumpf for A_Automaten_Typ;
    state A1, A2;

    initialize
        to A1
        begin
            ...
        end;

    trans
    from A1
        when X.x1 to A2
        begin
            ...
            output Y.y1;
        end;

    from A2
        when Y.y2 to A1
        begin
            ...
            output X.x2;
        end;

end;
```

Ausgaben an einen anderen Automaten werden mittels der output-Anweisung ausgedrückt, die im Beispiel jeweils am Ende der Transitionsblöcke von A stehen. Die drei Punkte vorher sollen andeuten, daß hier noch andere Aktionen, wie z.B. die Änderung des Wertes einer Kontexvariablen, stehen können.

Die Spezifikation des Automaten B hat folgende Form:

```
body B_Rumpf for B_Automaten_Typ;
    state B1, B2, B3;

    initialize
        to B1
        begin
            ...
        end;

    trans
```

```
    from B1
        when Y.y1 to B2
        begin
            ...
            output Z.z1;
        end;

    from B2
        when Z.z0 to same
        begin
            ...
        end;
        when Z.z4 to B3
        begin
            ...
            output Z.z3;
        end;

    from B3
        when Z.z5 to B1
        begin
            ...
            output Y.y2;
        end;
end;
```

Bisher wurden nur die Kommunikations-Schnittstellen und das Verhalten der Automaten A und B beschrieben, nicht aber der Gesamtaufbau des Kommunikationsmodells. Alle bisherigen Vereinbarungen werden in einem Spezifikationsrahmen zusammengefasst, in dem auch das Kommunikationsmodell vollständig definiert wird. Muß noch zusätzlich mit der Umgebung, z.B. einem Benutzer an einem Computer-Terminal, kommuniziert werden, so ist dies in Estelle nur durch die Benutzung von sogenannten primitiven Prozeduren möglich. In Estelle selbst wird nur der Prozedurkopf, d.h. der Name der Prozedur und ihre Parameter, spezifiziert. Die Prozedur wird in einer anderen Programmiersprache, meist C oder C++, geschrieben. Alle von diesen Programmiersprachen angebotenen Möglichkeiten, z.B. Ein-/Ausgabe auf Terminal, können in den Prozeduren benutzt werden.

```
specification Beispiel;

    Kanal-Definitionen
    Modul-Kopf und -Rumpf Definitionen
```

```
    modvar
        A : A_Automaten_Typ;
        B : B_Automaten_Typ;

    initialize
    begin
        init A with A_Rumpf;
        init B with B_Rumpf;
        ...
        connect A.y to B.y;
        ...
    end;
end;
```

4 Produktautomat

Die im vorherigen Kapitel vorgestellte Methode, mit der Automaten durch Estelle-Module spezifiziert werden können, läßt sich auch auf die Schichten des OSI-Basis-Referenz-Modells anwenden. Dies führt aber zu Laufzeit-ineffizienten Implementierungen, da bei dieser Spezifikationsweise jede Schicht isoliert von ihren Nachbarschichten betrachtet wird und die Kosten der Kommunikation zwischen den Schichten nicht berücksichtigt werden. Die vorhandenen Entwicklungswerkzeuge sind nicht in der Lage aus dieser Art der System-Spezifikation, welche sich auf einem hohen Abstraktionsniveau befindet, auf dem die Effizienzgesichtspunkte keine Rolle spielen, eine effiziente Implementation zu generieren. Eine einfache Implementierungsweise benutzt z.B. für jede Schicht einen eigenen Automaten, der als eigenständiger Prozeß auf dem Rechner abläuft. Untereinander kommunizieren die Automaten durch Interprozeßkommunikation. Diese Kommunikation hat einen relativ hohen Aufwand. Reicht nun eine Schicht, wie z.B. die Darstellungsschicht (siehe Seite 15), viele Dienste der darunterliegenden Schicht nur nach oben durch, so wird der Aufwand hierfür unverhältnismäßig groß und trägt entscheidend zum Gesamtaufwand der Protokollabwicklung bei.

Durch eine Verfeinerung der Spezifikation innerhalb der FDT Estelle ist es möglich Laufzeit-effizientere Implementationen zu erhalten. Eine Möglichkeit der Verfeinerung liegt im Verschmelzen benachbarter Schichten in einen Automaten. Werden zwei benachbarte Schichten miteinander verschmolzen, so spricht man von der Bildung des Produktautomaten dieser beiden Schichten. Bernd Hofmann von der Universität Mannheim hat dies für die Schichten 5 und 6 durchgeführt. Eine ausführliche Beschreibung seiner Arbeit findet sich in [Hof93].

Der Produktautomat wurde darin wie folgt definiert:

Seien $\alpha_j = (Z_j, E_j, A_j, \delta_j, \lambda_j, i_j), j \in \{1, 2\}$ zwei endliche Automaten mit den Zustandsmengen Z_j, den Eingabealphabeten E_j mit $E_1 \cap E_2 = 0$, den Ausgabealphabeten A_j, den Zustandsübergangsfunktionen $\delta_j : Z_j \times E_j \to Z_j$, den Ausgabefunktionen $\lambda_j : Z_j \times E_j \to A_j$ sowie den Startzuständen i_j. Dann ist der Produktautomat $\alpha_1 \times \alpha_2$ definiert durch :

$$\alpha_1 \times \alpha_2 := (Z_1 \times Z_2, E_1 \cup E_2, A_1 \cup A_2, \delta, \lambda, (i_1, i_2))$$

mit

$$\delta : (Z_1 \times Z_2) \times (E_1 \cup E_2) \to (Z_1 \times Z_2)$$

$$\delta((z_1, z_2), e) := \begin{cases} (\delta_1(z_1, e), z_2) & : \quad e \in E_1 \\ (z_1, \delta_2(z_2, e)) & : \quad e \in E_2 \end{cases}$$

und

$$\lambda : (Z_1 \times Z_2) \times (E_1 \cup E_2) \to (A_1 \cup A_2)$$

mit

$$\lambda((z_1, z_2), e) := \begin{cases} (\lambda_1(z_1, e), z_2) & : \quad e \in E_1 \\ (z_1, \lambda_2(z_2, e)) & : \quad e \in E_2 \end{cases}$$

4.1 Beispiel

Für das Beispiel aus Kapitel 3 bedeutet dies, daß die beiden Automaten A und B durch einen Automaten C, ihren Produktautomaten, ersetzt werden. An den Schnittstellen x und z weist dieser Automat das gleiche Verhalten auf wie die beiden kommunizierenden Automaten A und B, siehe Abbildung 8. Die Architektur des Automaten C ist in Abbildung 11 dargestellt.

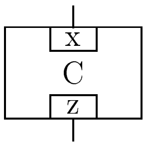

Abbildung 11: Architektur des Produktautomaten

Das Zustandsdiagramm des Automaten erhält man durch Anwendung der Definition des Produktautomaten auf die Automaten A und B. In Abbildung 12 ist das Zustandsdiagramm des Produktautomaten C dargestellt.

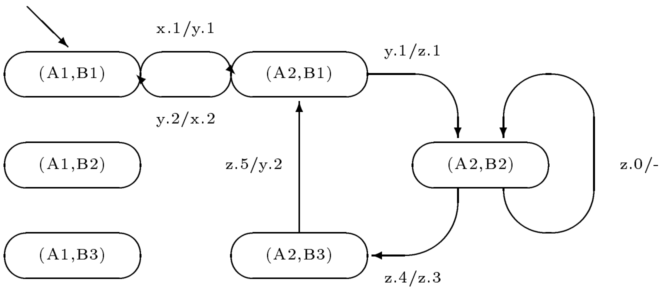

Abbildung 12: Zustandsdiagramm des Produktautomaten

Dieser Automat enthält noch Zustände $((A1, B2), (A1, B3))$, die nie erreicht werden, sowie Transitionen, die aufeinanderfolgend ausgeführt werden, d. h. ohne daß weitere äußere Ereignisse auftreten müssen oder eine beliebige andere Transition ungleich der zweiten schalten kann. Das ist genau dann der Fall, wenn das Ausgabezeichen der ersten Transition das Eingabezeichen der zweiten ist (z.B. Transition x.1/y.1 und Transition y.1/z.1). Solche Paare von Transitionen können zu einer Transition zusammengefaßt werden. In Spezialfällen kann durch diese Vereinfachung das Verhalten an den Schnittstellen des Produktautomaten von dem der beiden Einzel-Automaten abweichen. Falls einer der Einzel-Automaten in einem Zustand von beiden Eingangskanälen Transitionen entgegennehmen kann und ein Kanal bevorzugt behandelt wird, so ist diese

Prioritätenregelung im Produktautomaten möglicherweise nicht mehr vorhanden. Im Produktautomaten werden immer die Transitionen des mitintegrierten Automaten bevorzugt bearbeitet. Im Beispiel der Automaten A und B tritt dieser Fall aber nicht auf, ebensowenig in den OSI-Schichten 5 und 6. Eine formale Beschreibung der Umformung zur Vereinfachung des Produktautomaten findet sich in dem bereits erwähnten Bericht [Hof93].

Durch Anwendung dieser Transformation, sowie dem Entfernen nicht erreichbarer Zustände, ergibt sich für den Beispielautomaten C das Zustandsdiagramm in Abbildung 13.

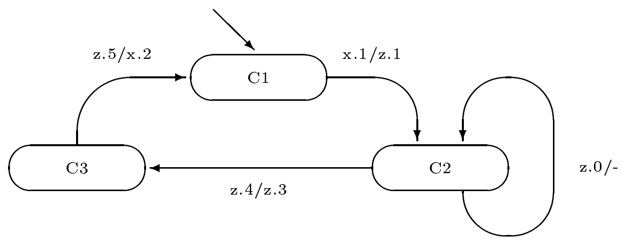

Abbildung 13: Transformiertes Zustandsdiagramm

Die Zustände wurden umbenannt, und es gilt die in Tabelle 8 dargestellte Zuordnung.

Produktzustand	Zustandsname
(A1, B1)	C1
(A2, B2)	C2
(A2, B3)	C3

Tabelle 8: Zustands-Bezeichnung

Dieser Automat kann nun in die Estelle-Spezifikation aus Kapitel 3 übernommen werden und die Automaten A und B ersetzen. Die Spezifikation des Automaten erfolgt in der dort vorgestellten Weise:

Der Modul-Kopf wird folgendermaßen spezifiziert:

```
module C_Automaten_Typ;
    ip X : xKanal(Anbieter);
       Z : zKanal(Benutzer);
end;
```

Der Modul-Rumpf ist definiert als:

```
body C_Rumpf for C_Automaten_Typ;
    state C1, C2, C3;

    initialize
        to C1
        begin
            ...
        end;

    trans
    from C1 to C2
        when X.x1
        begin
            ...
            output Z.z1;
        end;

    from C2
        when Z.z0 to same
        begin
            ...
        end;
        when Z.z4 to C3
        begin
            ...
            output Z.z3;
        end;

    from C3 to C1
        when Z.z5
        begin
            ...
            output X.x2;
        end;
end;
```

Der Gesamtaufbau des Kommunikationsmodells vereinfacht sich durch die Ersetzung von A und B durch C folgendermaßen:

```
specification Beispiel;

    Kanal-Definitionen
    Modul-Kopf und -Rumpf Definitionen

    modvar
        C : C_Automaten_Typ;

    initialize
    begin
        init C with C_Rumpf;
        ...

        ...
    end;
end;
```

Ein großer Nachteil bei der Verschmelzung von Schichten durch den Produktautomaten liegt in den hohen Kosten bei der Erstellung der Spezifikation. Die Bildung des Produktautomaten aus den Automaten der einzelnen Schichten, die Transformationen zur Vereinfachung des entstehenden Automaten und die Spezifikation dieses Automaten durch ein Estelle-Modul haben einen relativ hohen Arbeitsaufwand. Das Verfahren ist auch für mehr als zwei Schichten anwendbar, allerdings kann immer nur der Produktautomat von zwei Automaten in einem Schritt gebildet werden. Für n zu integrierende Automaten bedeutet dies die (n–1)-malige Bildung eines Produktautomaten. Der entstehende Produktautomat ist unlesbar, d.h. die integrierten Automaten lassen sich daraus nicht wiedererkennen, da sie nicht lokal auf einen Teil des Produktautomaten begrenzt sind. Dadurch muß bei der kleinsten Änderung in einem der beiden Ausgangsautomaten der Produktautomat völlig neu berechnet werden, da schon die geringste Änderung in einem Automaten das Aussehen des Produktautomaten verändern kann.

5 Strukturierter Produktautomat

Eine andere Möglichkeit einer effizienteren Implementation als durch die Bildung des Produktautomaten mehrerer Schichten, ist ihre Integration in einen strukturierten Produktautomaten. Dabei bleiben die Schichten auch weiterhin getrennt, sie werden aber innerhalb eines Estelle-Automaten abgewickelt. Durch diese an den Schichten orientierte Untergliederung der Spezifikation ist es möglich eine gut erkennbare 1-zu-1 Zuordnung zwischen Teilen der Spezifikation und den in Kapitel 3 vorgestellten Estelle-Konstrukten zur Beschreibung von Automatenverhalten herzustellen. Die Methode wurde von Jan Bredereke entwickelt und realisiert den Austausch von Interaktionen zwischen den integrierten Schichten durch Prozedur-Aufrufe. Dies ist aber mit dem Modul-Konzept von Estelle nicht möglich. Die Spezifikation der Schichten als Automaten muß daher auf eine andere Art und Weise geschehen.

Die Spezifikation der Automaten der einzelnen Schichten erfolgt durch eine Kombination von Prozeduren und Datenstrukturen, die das Verhalten und den Zustandsraum des jeweiligen Automaten beschreiben. In einem Record werden der aktuelle Zustand des Automaten sowie alle Kontextvariablen zusammengefaßt. Für jedes eingehende Datenpaket ist genau eine Prozedur, im folgenden Text als Verarbeitungs-Prozedur dieses Paketes bezeichnet, vorhanden. Diese Prozedur nimmt das Datenpaket entgegen und entscheidet in Abhängigkeit von dem aktuellen Zustand und den Werten der Kontextvariablen, welche Aktionen ausgeführt werden sollen. Folgende Aktionen sind dabei möglich:

- Ändern des aktuellen Zustands und der Werte der Kontextvariablen
- Aufruf weiterer Prozeduren, z.B. zur Umwandlung von Daten von einer Darstellung in eine andere
- Kommunikation mit anderen Automaten

Der Aufbau der Verarbeitungs-Prozeduren ist analog zu den FROM-Klauseln aus Kapitel 3. Dadurch ergibt sich für die Verarbeitungs-Prozedur jedes Paketes eine sehr einfache Struktur. Sie besteht aus einem CASE-Statement, wobei für jede mögliche Kombination von Zustand und Wert der Kontextvariablen die entsprechenden Aktionen ausgeführt werden.

Die Spezifikation des strukturierten Produktautomaten besteht aus den Verarbeitungs-Prozeduren und Datenstrukturen der integrierten Schichten sowie der Spezifikation eines normalen Automaten in Estelle. Die Aufgabe dieses Automaten ist die Verteilung der eingehenden Interaktionen an die dafür zuständigen integrierten Automaten. Diese Verteilung erfolgt durch einen Prozeduraufruf der jeweiligen Verarbeitungs-Prozedur. Der Automat benötigt zur Abwicklung seiner Aufgabe nur einen Zustand, da er ständig bereit ist, eingehende Interaktionen weiterzuleiten.

Bei der Kommunikation eines integrierten Automaten mit einem anderen beliebigen Automaten muß man zwei völlig verschiedene Fälle unterscheiden:

1. Der Kommunikationspartner ist eine andere in diesem Produktautomaten integrierte Schicht. Der Austausch der Daten wird durch den Prozeduraufruf der jeweiligen Verarbeitungs-Prozedur des Automatens der Empfänger-Schicht realisiert, wobei die Daten als Aufruf-Argument übergeben werden.

2. Der Kommunikationspartner ist eine Schicht, die nicht in dem strukturierten Produktautomaten integriert wurde. Der Austausch der Daten sollte, wie in Kapitel 3 beschrieben, durch die Estelle-Anweisung output realisiert werden. Dies ist in Estelle aber nicht möglich. Die output-Anweisung darf nur innerhalb eines Transitionsblocks stehen, aber nicht innerhalb einer Prozedur.

Zur Spezifikation eines strukturierten Produktautomaten ist also eine Erweiterung der Definition von Estelle nötig. In [Bre93] wurde von Jan Bredereke eine Erweiterung von Estelle vorgeschlagen, die unter anderem die output-Anweisung in Prozeduren zuläßt. In dem Bericht wurden ausführlich die dafür notwendigen Änderungen an Syntax und Semantik von Estelle diskutiert. Weiterhin wurden die sich aus den Änderungen ergebenden Konsequenzen und Möglichkeiten in Bezug auf die Spezifikation von Kommunikationsmodellen erläutert. Die Syntax- und Semantik-Änderungen lassen sich problemlos an vorhandenen Estelle-Werkzeugen durchführen. Eine solche Estelle-Entwicklungsumgebung, der pet-dingo-Compiler, der von Jan Bredereke entsprechend angepaßt worden war, stand für diese Projektarbeit zur Verfügung.

5.1 Beispiel

Anhand eines Beispiels wird die Spezifikationsweise des strukturierten Produktautomaten verdeutlicht. Dabei werden einige Stilregeln festgelegt, die eine mehr oder weniger mechanische Erstellung der Spezifikation des strukturierten Produktautomaten ermöglichen. Dadurch soll auch eine Rückübersetzung der Spezifikation in eine Spezifikation nach herkömmliche Spezifikationsweise leicht durchführbar bleiben. Die einzelnen Stilregeln werden im folgenden Text besonders hervorgehoben.

Will man die beiden miteinander kommunizierenden Automaten des Beispiels eines Kommunikationsmodells aus Kapitel 3 in einem Automaten integrieren, indem man den strukturierten Produktautomaten der beiden Automaten bildet, so sind einige zusätzliche Typ-Vereinbarungen unerläßlich. Da die Verwaltung des Zustandes jedes der beiden Automaten nun nicht mehr durch Estelle-Module geregelt werden, sondern in Prozeduren explizit gehandhabt werden müssen, ist die **Deklaration von zwei Aufzählungstypen, welche als Wertebereich die symbolischen Zustandsbezeichnungen je eines der beiden Automaten haben**, empfehlenswert.

```
type
A_Zustands_Typ = ( A1 , A2 );
B_Zustands_Typ = ( B1, B2, B3 );
```

Wie weiter oben erwähnt, sollten **der Zustand und die Kontextvariablen eines Automaten in einem Record zusammengefaßt** werden. Dieser Record wird im folgenden als Verwaltungs-Record des Automaten bezeichnet. Zum einen sind in dieser Struktur alle zu einem Automaten gehörenden Daten zusammengefaßt, was spätere Veränderungen an dem Automaten erleichtert, zum anderen beschleunigt dies auch den Ablauf, da den einzelnen Verarbeitungs-Prozeduren nur eine Referenz auf diesen Record übergeben werden kann und dies nicht für jede einzelne Variable geschehen muß.

In dem Beispiel besitzen zwar beide Automaten keine Kontextvariablen und deshalb wären zur Speicherung der Daten der Automaten keine Verwaltungs-Records nötig. Zur Verdeutlichung der allgemeinen Vorgehensweise wird der Zustand der Automaten aber in der Komponente „Zustand“ des jeweiligen Verwaltungs-Records gespeichert. Dadurch ergeben sich folgende Definitionen der Records:

```
A_Verwaltungs_Typ = record
      Zustand : A_Zustands_Typ;
end;
B_Verwaltungs_Typ = record
      Zustand : B_Zustands_Typ;
end;
```

Für jede mögliche eingehende Interaktion müssen Prozeduren vorhanden sein, die diese verarbeiten. Diese Prozeduren bekommen als Parameter die eingegangene Interaktion sowie den Verwaltungs-Record übergeben und führen die entsprechenden Aktionen zu dieser Interaktion aus. Innerhalb der Verarbeitungs-Prozeduren kann mittels der WITH-DO-Anweisung mit den selben Bezeichnern auf die Kontextvariablen zugegriffen werden wie in der herkömmlichen Spezifikationsweise.

Der Automat A nimmt zwei verschiedene Interaktionstypen entgegen:

- x.1
- y.2

Die entsprechenden Verarbeitungs-Prozeduren sehen folgendermaßen aus:

```
procedure A_Verarbeite_X1(var V_Record : A_Verwaltungs_Typ; var paket : X1);
var y1 : Y1;
begin
    with V_Record do begin
        case Zustand of
            A1 : begin
                B_Verarbeite_Y1(B_V_Record, y1);
                Zustand := A2;
            end;
            A2 : begin
                Fehler(Zustand, paket);
            end;
        end; (* of case *)
    end; (* of with *)
end;

procedure A_Verarbeite_Y2(var V_Record : A_Verwaltungs_Typ; var paket : Y2);
begin
    with V_Record do begin
        case Zustand of
            A2 : begin
                output X.x2;
                Zustand := A1;
            end;
            A1 : begin
                Fehler( Zustand, paket);
            end;
        end; (* of case *)
    end; (* of with *)
end;
```

An diesen beiden Verarbeitungs-Prozeduren zeigt sich deutlich der Unterschied zwischen der Kommunikation mit einem ebenfalls in diesen strukturierten Produktautomaten integrierten Automaten und mit einem nicht darin integrierten Automaten.

Die Interaktion y1, die an den Automaten B gesendet werden soll, wird mittels des Aufrufes der Verarbeitungs-Prozedur B_Verarbeite_Y1 des Automaten B mit y1 als Argument abgehandelt. Als weiteres Argument wird der Verwaltungs-Record des Automaten B mit übergeben. Durch diese Parameter soll eine eindeutige Zuordnung zwischen den Datenstrukturen und den Prozeduren, welche auf sie zugreifen, erreicht werden. Der Verwaltungs-Record des Automaten B muß also in den Verarbeitungs-Prozeduren von A sichtbar sein und umgekehrt. Dies erreicht man, indem die Variablen zur Speicherung der Verwaltungs-Records vor den Prozeduren vereinbart werden. Zusätzlich benötigt man noch forward-Deklarationen der Verarbeitungs-Prozeduren, da in den Verarbeitungs-Prozeduren des Automaten A Verarbeitungs-Prozeduren des Automaten B aufgerufen werden können und umgekehrt. Die Interaktion x2, die an einen hier nicht näher spezifizierten Automaten, der aber nicht in den strukturierten Produktautomaten integriert wurde, gesendet werden soll, wird mittels der Estelle-Anweisung output über den Kanal x gesendet.

Die Anweisungsblöcke mit dem Aufruf einer Prozedur „Fehler“ in den CASE-Anweisungen, z.B. der CASE-Fall A1 in der Prozedur A_Verarbeite_Y2, sind nötig, da in der case-Anweisung alle möglichen Fälle abgedeckt werden müssen. Auf das Automaten-Modell übertragen bedeutet ein solcher CASE-Fall einen unspezifizierten Empfang des Automaten A. Es sollte darauf entsprechend reagiert werden, z.B. eine Fehlerverarbeitungsprozedur aufgerufen werden. Im Beispiel wurde dies durch den Aufruf der Prozedur „Fehler“ gemacht. Diese Prozedur ist keine gültige Prozedur in Estelle-Syntax, da die Argumente von variablem Typ sind. Die Schreibweise wurde auf Grund der besseren Lesbarkeit benutzt, eine entsprechende Umsetzung der Prozedur in Estelle mittels mehrerer Prozeduren ist einfach zu erreichen.

Ein Zustandswechsel erfolgt durch die Zuweisung des neuen Zustandwertes an die Komponente „Zustand“ des Verwaltungs-Records. Da dieser als Referenzparameter übergeben wurde, bleibt die Änderung auch nach dem Verlassen der Prozedur gültig. **Diese Zuweisung steht immer am Ende des Anweisungsblocks**, es erleichtert so die Überprüfung, ob alle Zustandswechsel des Automaten auch in den Verarbeitungs-Prozeduren vorkommen.

Die Verarbeitungs-Prozeduren des Automaten B haben die gleiche Struktur wie die des Automaten A.

```
procedure B_Verarbeite_Y1(var V_Record : B_Verwaltungs_Typ; var paket : Y1);
begin
    with V_Record do begin
        case Zustand of
            B1 : begin
                output X.x2;
                Zustand := B2;
            end;
            B2, B3 : begin
                Fehler(Zustand, paket);
            end;
        end; (* of case *)
    end; (* of with *)
end;
```

```
procedure B_Verarbeite_Z0(var V_Record : B_Verwaltungs_Typ; var paket : Z0);
begin
    with V_Record do begin
        case Zustand of
            B2 : begin
            end;
            B1, B3 : begin
                Fehler(Zustand, paket);
            end;
        end; (* of case *)
    end; (* of with *)
end;

procedure B_Verarbeite_Z4(var V_Record : B_Verwaltungs_Typ; var paket : Z4);
begin
    with V_Record do begin
        case Zustand of
            B2 : begin
                output Z.z3;
                Zustand := B3;
            end;
            B1, B3 : begin
                Fehler(Zustand, paket);
            end;
        end; (* of case *)
    end; (* of with *)
end;

procedure B_Verarbeite_Z5(var V_Record: B_Verwaltungs_Typ; var paket : Z5);
var y2 : Y2; begin
    with V_Record do begin
        case Zustand of
            B3 : begin
                A_Verarbeite_Y2(A_V_Record, y2);
                Zustand := B1;
            end;
            B1, B2 : begin
                Fehler(Zustand, paket);
            end;
        end; (* of case *)
    end; (* of with *)
end;
```

Der strukturierte Produktautomat hat den gleichen äußeren Aufbau wie der Produktautomat aus Kapitel 4, siehe Abbildung 11. Das Estelle-Modul, welches den strukturierten Produktautomaten der beiden Automaten A und B repräsentiert, ist sehr einfach aufgebaut. Als Schnittstellen zu anderen Modulen hat es die beiden Kanäle X und Z. Das Modul benötigt auch nur einen Zustand, um seine Aufgabe abzuwickeln. In diesem Zustand ist es immer bereit, eingehende Interaktionen entgegenzunehmen und die entsprechenden Verarbeitungsprozeduren aufzurufen. In den Transitionsblöcken für die einzelnen Interaktionen stehen keine output-Anweisungen, da Ausgaben innerhalb der Verarbeitungs-Prozeduren geschehen. Da in den Verarbeitungs-Prozeduren auch die Zustandsübergänge durchgeführt werden, steht in den einzelnen Transitionsblöcken nur der Aufruf der jeweiligen Verarbeitungs-Prozedur.

Das Modul kann folgendermaßen spezifiziert werden, wobei bereits bekannte Teile durch einen symbolischen Namen repräsentiert werden.

Die Modulkopf-Spezifikation für den strukturierten Produktautomaten kann direkt von der des Produktautomaten aus Kapitel 4 übernommen werden. Es muß nur ein anderer Modul-Rumpf dafür spezifiziert werden. Dieser sieht für den strukturierten Produktautomaten wie folgt aus:

```
body D_Rumpf for C_Automaten_Typ;
    var A_V_Record : A_Verwaltungs_Typ;
        B_V_Record : B_Verwaltungs_Typ;

    Verarbeitungs-Prozeduren

    initialize
    begin
        A_V_Record.Zustand := A1;
        B_V_Record.Zustand := B1;
    end;

    trans
    when X.x1(p1)
    begin
        A_Verarbeite_X1(A_V_Record, p1);
    end;

    when Z.z0(p0)
    begin
        B_Verarbeite_Z0(B_V_Record, p0);
    end;

    when Z.z4(p4)
    begin
```

```
        B_Verarbeite_Z4(B_V_Record, p4);
    end;

    when Z.z5(p5)
    begin
        B_Verarbeite_Z5(B_V_Record, p5);
    end;
end;
```

Für den strukturierten Produktautomaten wurden, wie man an der fehlenden state-Anweisung sieht, keine expliziten Zustände definiert. Damit besitzt er implizit genau einen Zustand. In dem INITIALIZE-Anweisungsblock werden den Komponenten der Verwaltungs-Records, welche die Zustände der beiden integrierten Automaten beinhalten, definierte Startwerte zugewiesen und damit eine Initialisierung der Automaten A und B vorgenommen.

Die Parameter der einzelnen Interaktionen, z.B. p1, sind die Daten, die während der Kommunikation zwischen den Automaten ausgetauscht werden. Diese Daten werden den Verarbeitungs-Prozeduren aus Geschwindigkeitsgründen als Referenzparameter übergeben.

Die Spezifikation der gesamten Kommunikationsarchitektur kann wiederum aus dem Kapitel über den Produktautomaten übernommen werden, nur die Initialisierung der Modulvariablen C muß, anstelle von Rumpf C_Rumpf des Produktautomaten, mit dem Modul-Rumpf D_Rumpf des strukturierten Produktautomaten erfolgen.

Im Laufe der Lebenszeit eines Software-Produkts müssen üblicherweise Änderungen an der Spezifikation vorgenommen werden, z.B. um die Spezifikation in ihrer Funktionalität zu erweitern. Deshalb sollte die Spezifikation möglichst gut strukturiert sein, damit sich Änderungen nur lokal auswirken und damit leicht durchzuführen sind. Die Spezifikationsweise des strukturierten Produktautomaten erfüllt diese Anforderung. Änderungen an einem der beiden integrierten Automaten beschränken sich lokal auf seine Verarbeitungs-Prozeduren. Die Prozeduren des anderen integrierten Automaten bleiben davon unberührt. Das Estelle-Modul selbst ändert sich nur, wenn sich an seinen Schnittstellen zur Außenwelt etwas ändert.

Gegenüber der Spezifikation des Produktautomaten sind Veränderungen hier also sehr viel einfacher durchzuführen und erfordern nicht eine komplette Neuspezifikation. Der Aufwand zur Erstellung der Spezifikation ist beim strukturierten Produktautomaten auch geringer, da keine komplizierte Transformation von Automaten durchgeführt werden muß wie bei der Bildung des Produktautomaten. Man kann sich vielmehr an den vorhandenen Spezifikationen der zu integrierenden Automaten orientieren und die Verarbeitungs-Prozeduren entsprechend aufbauen.

6 Integration der Schichten 5 und 6 in einem Automaten

Die Integration von zwei Schichten durch die Bildung des strukturierten Produktautomaten wurde an der Sitzungs- und Darstellungs-Schicht des OSI-Schichtenmodells durchgeführt. Ausgehend von einer herkömmlichen Spezifikation dieser beiden Schichten, d.h. also ein Estelle-Modul pro Schicht, wurde diese Spezifikation schrittweise so verändert, bis beide Schichten durch einen Automaten spezifiziert wurden. In diesem Kapitel werden die notwendigen Änderungen und die Umformungsschritte, die zur Transformation in den strukturierten Produktautomaten benötigt wurden, beschrieben.

In der herkömmlichen Spezifikation werden primitive Prozeduren benutzt. Diese Prozeduren sind in C geschrieben und realisieren das Umpacken von Daten aus einem Pakettyp in einen anderen. Die in dieser Projektarbeit untersuchten Spezifikationen sollten keine primitiven Prozeduren mehr enthalten, da so eine Unabhängigkeit von den verwendeten Entwicklungswerkzeugen erreicht wird. Da fast jedes Entwicklungswerkzeug eigene Methoden zur Einbindung primitiver Prozeduren benutzt, sind oft Änderungen an den Spezifikationen beim Wechsel des Entwicklungswerkzeugs nötig.

Der Verzicht auf primitive Prozeduren in der Spezifikation führte dazu, daß an mehreren Stellen in der herkömmlichen Estelle-Spezifikation Anpassungen vorgenommen werden mußten. Im Anhang A werden für die einzelnen Quelltext-Dateien die wichtigsten Änderungen erklärt.

6.1 Vorgehensweise

Die Transformation von der herkömmlichen Spezifizierung zur Integration mittels strukturiertem Produktautomaten wurde schrittweise vorgenommen. Dabei entstand nach jedem Transformationsschritt eine lauffähige Spezifikation. Diese konnte dann getestet werden und so das Auftreten von Fehlern schon früh erkannt und die Fehlerquelle lokal stark begrenzt werden. Die Umformung geschah in vier Schritten, die im folgenden kurz erläutert werden.

1. Alle Kontextvariablen der zu integrierenden Automaten wurden in jeweils einem Record zusammengefaßt.

2. Zu jeder ausgehenden Interaktion wurde eine eigene Ausgabeprozedur definiert und die output-Anweisungen in den Transitionsblöcken durch Prozeduraufrufe der jeweiligen Ausgabeprozedur ersetzt. Dadurch, daß die gleiche Interaktion in verschiedenen Transitionsblöcken auftrat, das Belegen der einzelnen Datenfelder der Interaktion aber bei allen diesen Transitionsblöcken sehr ähnlich ist, konnte durch die Zusammenfassung

zu einer Ausgabeprozedur der Quelltext kompakter gestaltet werden, ohne an Übersichtlichkeit zu verlieren. Dieser Schritt erleichtert auch die endgültige Transformation in einen strukturierten Produktautomaten durch Schritt 4, da ein Teil der bei diesem Schritt noch vorzunehmenden Änderungen in den Ausgabeprozeduren gebündelt ist.

3. Die zu integrierenden Automaten wurden nun durch eine Datenstruktur mit allen für das Verhalten des Automaten relevanten Daten und Verarbeitungs-Prozeduren für jede eingehende Interaktion spezifiziert, wie dies am Beispiel der beiden kommunizierenden Automaten in Kapitel 5 vorgeführt wurde. Der Record aus Schritt 1 wurde dazu bei jedem der zu integrierenden Automaten um eine Komponente erweitert, die den Zustand des Automaten repräsentiert.

4. Im letzten Schritt wurden die Records, die Ausgabeprozeduren und Verarbeitungs-Prozeduren der zu integrierenden Automaten in einem Estelle-Modul zusammengefaßt, wie dies in Kapitel 5 demonstriert wurde. Alle output-Anweisungen in den Ausgabeprozeduren, die an einen ebenfalls mitintegrierten Automaten gerichtet waren, wurden durch Aufrufe der jeweiligen Verarbeitungs-Prozedur ersetzt. Das Estelle-Modul enthält nur einen Zustand, in dem es alle für die integrierten Automaten bestimmten Interaktionen entgegen nimmt und die jeweiligen Verarbeitungsprozeduren aufruft, wie dies schon in Kapitel 5 beschrieben wurde.

Diese Vorgehensweise zur Transformation wurde zweimal angewandt. Zuerst wurde die Spezifikation der Sitzungs-Schicht, die in der herkömmlichen Spezifikation aus zwei getrennten Automaten bestand, in einem Automaten zusammengefaßt. Aus dieser Spezifikation und der Spezifikation der Darstellungsschicht wurde dann der strukturierte Produktautomat gebildet. Eine weitere Integration von Nachbarschichten, z.B. der Anwendungs-Schicht, ist problemlos möglich.

7 Messungen

Damit man die einzelnen Spezifikations-Arten im Hinblick auf Effizienz bei der Ausführung der Protokolle miteinander vergleichen kann, wurden eine Reihe von Messungen durchgeführt. Jede der Spezifikations-Arten der Schichten 5 und 6, welche in den vorherigen Kapiteln beschrieben wurden, wurde in die gleiche Testumgebung eingesetzt und die Laufzeit für die Übertragung von Datenpaketen gemessen. Der Aufbau der Testumgebung garantierte dabei, daß alle Messungen unter den gleichen Bedingungen abliefen, d.h. z.B., daß die Messungen unabhängig von der Auslastung des Rechners waren.

7.1 Die Test-Umgebung

Die verschiedenen Spezifikations-Arten der Schichten 5 und 6 wurden in eine Testumgebung eingebettet, deren Aufbau in Abbildung 14 wiedergegeben ist.

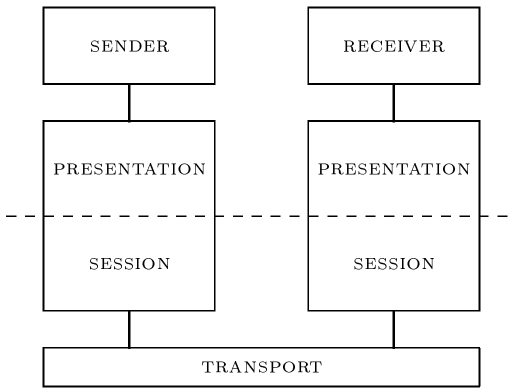

Abbildung 14: Testumgebung

Der Testablauf stellt eine typische Benutzung der Dienste der einzelnen Schichten dar, die Übertragung von Daten von einem Sender zu einem Empfänger. Der Sender muß dabei eine Verbindungsanforderung stellen und der Empfänger diesen Aufbauwunsch positiv bestätigen. Nach erfolgreichem Aufbau der Verbindung überträgt der Sender die Daten, während der Empfänger die eingehenden Datenpakete nur entgegennimmt, selbst aber keine Daten verschickt. Hat der Sender alle Daten übertragen, fordert er den Abbau der Verbindung an, und der Empfänger bestätigt den Abbauwunsch positiv.

In der Abbildung 15 ist der Zustandsgraph eines Automaten abgebildet, welcher das oben geforderte Verhalten eines Senders realisiert.

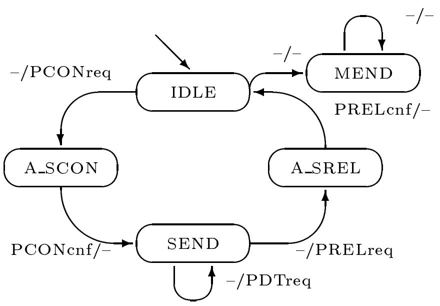

Abbildung 15: Sender-Automat

In dem Transitionsdiagramm in Abbildung 15 wurden keine Kontextvariablen berücksichtigt. Deshalb gibt es für einige Zustände mehrere mögliche Folgezustände. In der Estelle-Spezifikation des Automaten ist dieser Indeterminismus durch die Kontextvariablen aufgelöst. Mittels der Kontextvariablen lassen sich die Anzahl der zu übertragenden Datenpakete pro Durchlauf, sowie die Anzahl der Durchläufe, d.h. der Folge Verbindungsaufbau, Datenübertragung und Verbindungsabbau, bestimmen.

Das Transitionsdiagramm eines Automaten, welcher als Receiver in die Testumgebung eingesetzt wurde, zeigt Abbildung 16.

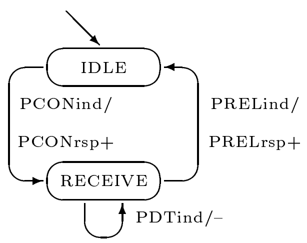

Abbildung 16: Receiver-Automat

Dieser Automat ist einfacher aufgebaut als der Sender-Automat. In Start-Zustand IDLE besteht keine Verbindung zu einem Sender. Der Receiver bestätigt jeden eingehenden Verbindungsaufbauwunsch (PCONind) eines Senders positiv (PCONrsp+) und geht in den Zustand RECEIVE über. In diesem Zustand empfängt der Automat alle eingehenden Datenpakete (PDTind) bis ein Verbindungsabbauwunsch (PRELind) eintrifft, der positiv bestätigt wird (PRELrsp+). Nach Abbau der Verbindung befindet sich der Receiver-Automat wieder im Zustand IDLE.

Da die Verbindungsaufbau-Transition im Sender nicht von externen Ereignissen, wie z. B. eingehenden Interaktionen oder Benutzereingaben, abhängig ist, laufen die Datenübertragungen sofort nach dem Start des gesamten Kommunikationssystems an, ohne daß noch irgendwelche Aktionen von Seiten des Benutzers nötig wären. Für die Messungen wurde die Zeit, die in den Modulen SENDER, TRANSPORT und RECEIVER beim Ablauf des Test verbraucht wurde, gleich 0 angenommen.

Die Testumgebung lief innerhalb eines Unix-Prozesses auf einer SPARCstation 10 der Firma Sun. Da es sich bei dieser Maschine um ein Einprozessorsystem handelt, ergibt eine Verteilung der Testumgebung auf mehrere Prozesse, z.B. je ein Prozeß pro Estelle-Modul, keine Geschwindigkeitsvorteile gegenüber der benutzten Implementation. Mittels des System-Befehls clock wurde die Laufzeit zur Übertragung einer Anzahl Datenpakete von Sender zu Receiver gemessen. Der Systembefehl clock gibt die bisherige verbrauchte Rechenzeit des Prozesses zurück. Direkt vor der Verbindungsaufbau-Anforderung des Senders wurde clock aufgerufen und der Rückgabewert gespeichert. Nachdem die Daten übertragen waren und die Verbindung wieder abgebaut war, erfolgte ein erneuter Aufruf von clock. Die Differenz zwischen Rückgabewert und dem gespeicherten Wert ist die Zeit, die benötigt wurde zur Abwicklung des gesamten Protokolls.

7.2 Aufbau der Meßreihen

Im folgenden wird die Spezifikation, welche als Grundlage für die Transformationen in andere Spezifikations-Arten diente, als Referenz-Spezifikation bezeichnet. Diese Spezifikation realisiert die Anwendungs-Schicht durch einen Automaten und die Sitzungs-Schicht durch zwei, wobei einer der beiden Sitzungs-Schicht-Automaten für das Verpacken und das Entpacken von SPDUs in bzw. aus Datenpaketen der Transport-Schicht zuständig ist, während der andere die Kerndienste der Sitzungs-Schicht verarbeitet.

Eine Spezifikation mit nur je einem Automaten pro Schicht wurde durch Anwendung der Methode des strukturierten Produktautomaten auf die beiden Sitzungs-Schicht-Automaten der Referenz-Spezifikation gewonnen. Sie wird im Text als Session-Integration bezeichnet.

Die Spezifikation des strukturierten Produktautomaten der Schichten 5 und 6 wird im folgenden als strukt. Integration bezeichnet.

Die Spezifikation des Produktautomaten der OSI-Schichten 5 und 6 wurde von Bernd Hofmann freundlicherweise für Vergleichsmessungen zur Verfügung gestellt. Sie wird im Text als Produkt-Integration bezeichnet. Leider bietet dieser Automat nicht alle Dienstelemente der Kerndienste der beiden Schichten. Es ist nur möglich, einen Verbindungsaufbau zu initiieren, Daten zu übertragen und zu empfangen, sowie den geordneten Verbindungsabbau anzufordern. Auf eintreffende Verbindungsaufbau-/abbauwünsche reagiert der Produktautomat nicht, es kommt zum unspezifizierten Empfang. Deswegen kann man den Produktautomaten in der Testumgebung nur unterhalb des SENDER-Moduls, auf der Senderseite, einsetzen. Auf der Empfängerseite muß eine andere Implementation der beiden Schichten vorhanden sein, die zumindest einen Verbindungsaufbauwunsch positiv bestätigen, Daten empfangen und den geordneten Verbindungsabbau bestätigen kann. Bei den folgenden Messungen wurde auf der Empfängerseite immer die Referenz-Spezifikation eingesetzt, falls der Produktautomat getestet wurde.

Um die verschiedenen Spezifikations-Arten der OSI-Schichten 5 und 6 miteinander vergleichen zu können, wurden 2 Meßreihen mit jeweils unterschiedlichem Aufbau der Testumgebung durchgeführt. Bei der einen Meßreihe waren auf Sender– und Empfängerseite jeweils die gleiche Spezifikation. Bei der anderen war auf der Empfängerseite immer die Referenz-Spezifikation, während auf der Senderseite verschiedene Spezifikationen eingesetzt wurden.

Die in der Tabelle 9 aufgeführten Kombinationen von Spezifikationen wurden getestet.

Nr.	Senderseite	Empfängerseite
1)	Referenz	Referenz
2)	Session-Integration	Session-Integration
3)	strukt. Integration	strukt. Integration
4)	Produkt-Integration	Referenz
5)	strukt. Integration	Referenz

Tabelle 9: Meßreihen-Aufbau

Die Kombinationen 1,2,3 aus Tabelle 9 bilden die erste Meßreihe und 1,4,5 die zweite.

Die erzielten Meßergebnisse für beide Meßreihen befinden sich im Anhang B.

8 Ergebnisse

Zur Beurteilung der Leistungsfähigkeit der Methode des strukturierten Produktautomaten gegenüber den anderen hier vorgestellten Spezifikationsformen werden zwei Aspekte der Spezifikation näher betrachtet. Zum einen die Laufzeit-Effizienz der aus der Spezifikationsart entstanden Implementation und zum anderen der Aufwand, der zur Erstellung und zur Wartung der jeweiligen Spezifikation notwendig ist.

8.1 Laufzeit-Aufwand

Für den Vergleich der Spezifikationsarten im Hinblick auf die Laufzeit-Effizienz werden die Meßergebnisse aus dem Anhang herangezogen. Die Auflösung des Clock-Systemaufrufes ist begrenzt. Deshalb unterscheiden sich die Meßwerte auch nur in den ersten beiden Stellen nach dem Komma. Die Werte für die Varianz, die Standardabweichung und die relative Standardabweichung belegen einen sehr geringen statistischen Meßfehler in den Messungen, so daß also die verwendete Meßmethode ausgesprochen gute Ergebnisse liefert. Tabelle 10 enthält die Mittelwerte der Meßergebnisse für die unterschiedlichen Kombinationen von Spezifikationen der Schichten 5 und 6, die auf Sender- und Empfängerseite in der Testumgebung eingesetzt waren. Die Bezeichnungen der einzelnen Kombinationen stammen aus Abschnitt 7.2.

Kombination	1 Datenpaket	10 Datenpakete	100 Datenpakete
1)	0.26	0.42	1.95
2)	0.12	0.20	0.68
3)	0.07	0.10	0.49
4)	0.14	0.25	1.23
5)	0.14	0.25	1.26

Tabelle 10: Mittelwerte der Messungen in Sekunden

Die Meßergebnisse der Kombinationen 4) und 5) sind gleich, d.h. die Methode des strukturierten Produktautomaten erzeugt eine ebenso effiziente Implementation wie sie durch Bildung des Produktautomaten der beiden Schichten erzielt wird. Da die vorliegende Spezifikation eines Produktautomaten nicht alle Dienste der Schichten 5 und 6 realisierte, läßt sich vermuten, daß der strukturierte Produktautomat sogar schneller ist als der Produktautomat.

Vergleicht man die Ergebnisse von 4) und 5) mit den Werten von 1), d.h. den Werten der Referenz-Spezifikation, ergibt sich der in der Tabelle 11 Zeile 2 dargestellte Speedup.

	1 Datenpaket	10 Datenpakete	100 Datenpakete
1) / 1)	1	1	1
1) / 5)	1.86	1.68	1.54
1) / 2)	2.17	2.10	2.22
1) / 3)	3.71	4.20	3.98

Tabelle 11: Speedup

Man erkennt deutlich, daß der größte Geschwindigkeitsgewinn dann erzielt wird, wenn nur ein Datenpaket vom Sender zum Receiver übertragen wird, da dann alle beteiligten Dienste, Verbindungsaufbau, Datenübertragung und Verbindungsabbau, gleich stark zum Speedup beitragen. Da der Protokollaufwand beim Verbindungsaufbau, besonders in der Sitzungs-Schicht, sehr viel größer ist als der zur Übertragung eines Datenpaketes, kann dabei auch besonders viel Zeit durch effizientere Abarbeitung des Dienstes gewonnen werden. Daher nimmt der Speedup mit steigender Anzahl an gesendeten Datenpaketen ab. Die Messungen bestätigen damit den in [Hof93] angegebenen maximalen Speedup von 2.

Bei den Kombinationen 4) und 5) war auf der Empfänger-Seite die Referenz-Spezifikation eingesetzt. Deswegen wurde der maximal mögliche Speedup von 2 auch nicht erreicht, da die langsamere Implementation auf der Empfängerseite die Geschwindigkeit der Abarbeitung der Protokolle bestimmte.

Wird auf beiden Seiten der strukturierte Produktautomat benutzt, wie in Kombination 3), so ergeben sich die in Tabelle 11 Zeile 4 aufgelisteten Speedups.

Die Geschwindigkeitssteigerung nimmt nicht mehr mit wachsender Zahl an gesendeten Datenpaketen ab, wie in Zeile 2, sondern bleibt relativ stabil. Hier ist der Speedup ungefähr 4, d.h. die durch die Methode des strukturierten Produktautomaten erzielte Spezifikation erzeugt gegenüber der herkömmlichen Spezifikationsart eine 4-mal so schnelle Implementierung.

Allein durch die Integration der beiden Sitzungs-Schicht-Automaten in einem strukturierten Produktautomaten, ergibt sich bei Nutzung dieses Automaten auf Sender- und Empfängerseite, wie dies bei Kombination 2) der Fall ist, ein Speedup von 2 gegenüber der Referenz-Spezifikation. In Tabelle 11 Zeile 3 sind die Werte des Speedups für jeweils 1, 10 und 100 gesendete Datenpakete aufgelistet.

8.2 Spezifikations-Aufwand

Der Aufwand zur Spezifikation des strukturierten Produktautomaten mehrerer Schichten ist in etwa vergleichbar mit dem Aufwand zur Spezifikation der einzelnen Schichten mittels der herkömmlichen Methode, also jeweils ein Automat pro Schicht. Die Transformation von ISO-Norm Spezifikation (in Tabellenform) in Estelle-Module (bei herkömmlicher Spezifikationsweise) bzw. in ein Estelle-Modul mit den dazugehörigen Datenstrukturen und Prozeduren (bei dem strukturierten Produktautomaten), ist nach einem ähnlichen Schema zu vollziehen. Die Transformation einer bereits existierenden Spezifikation im herkömmlichen Stil in einen strukturierten Produktautomaten, wie sie in dieser Projektarbeit durchgeführt wurde, benötigt einen relativ geringen Arbeitsaufwand. Befolgt man die in Kapitel 5 vorgeschlagenen Transformationsschritte, so beschränkt sich fast jeder dieser Schritte auf das Kopieren von Teilen der vorhandenen Spezifikation in die entsprechenden Konstrukte des strukturierten Produktautomaten, z.B. das Kopieren eines Transitionsblocks in den entsprechenden CASE-Block der Verarbeitungs-Prozedur. Hat man die Rahmenkonstrukte des strukturierten Produktautomaten erstellt, also Verarbeitungsprozeduren, Verwaltungs-Records, Ausgabe-Prozeduren und den zentralen Estelle-Automaten, und die entsprechenden Teile aus der Original-Spezifikation herauskopiert, so sind nur noch wenige Änderungen nötig. Dies ist z.B. das teilweise Ersetzen der output-Anweisung durch Prozeduraufrufe der Verarbeitungs-Prozeduren. Insgesamt benötigt man für diese Transformation einer bereits vorhandenen Spezifikation in Estelle weniger Arbeitsaufwand als für die komplette Neuerstellung aus der ISO-Norm.

Die Lesbarkeit der Spezifikation des strukturierten Produktautomaten hat sich gegenüber der herkömmlichen Spezifikationsform nur geringfügig verschlechtert. Aktionen in den Transitionsblöcken der herkömmlichen Spezifikation werden in die Verarbeitungs- und Ausgabe-Prozeduren verlagert. Die Transitionsblöcke des strukturierten Produktautomaten enthalten nur noch die Prozeduraufrufe der entsprechenden Verarbeitungs-Prozeduren. Um nun die Aktionen festzustellen, die bei einer eingehenden Interaktion ausgeführt werden, muß in verschiedenen Prozeduren nachgeschaut werden, und nicht wie bei der herkömmlichen Spezifikationsweise an einer Stelle, nämlich im Transitionsblock. Die Kommunikation zwischen den integrierten Automaten ist nur durch genauere Untersuchung aller Ausgabe- und Verarbeitungs-Prozeduren erkennbar.

Es bleiben also in beiden Spezifikationstilen die einzelnen Automaten der Schichten klar von einander getrennt, so daß Änderungen an einer Schicht sich nur lokal auswirken und somit leicht zu bewerkstelligen sind.

Dies ist bei der Methode des Produktautomaten nicht der Fall. Damit mehrere Schichten in einem Automaten integriert werden können, muß die Bildung des Produktautomaten mehrmals angewandt werden. Zuerst wird der Produktautomat der Automaten zweier benachbarter Schichten gebildet, von diesem

Automaten und einem Automaten einer angrenzenden Schicht muß wiederum der Produktautomat gebildet werden. Dies wird solange fortgesetzt bis alle Schichten in einem einzigen Produktautomaten integriert sind. Die einzelnen Schichten sind in diesem Automaten nicht lokal auf bestimmte Teilbereiche des Produktautomaten beschränkt, sondern durch die wiederholte Produktbildung immer auf den gesamten Produktautomaten verteilt. Deswegen muß bei einer Änderung in einer Schicht der ganze Prozeß der Bildung der Produktautomaten von vorne begonnen werden.

Schon die Spezifikation eines Produktautomaten zweier Schichten in Estelle ist dabei deutlich aufwendiger als die Spezifikation dieser Schichten in zwei getrennten Automaten, bzw. in einem strukturierten Produktautomaten, da zuerst aus den beiden Automatenmodellen der Schichten der Produktautomat gebildet werden muß, dieser wird anschließend minimiert und dann erst in ein Estelle-Modul transformiert.

8.3 Fazit

Die Methode des strukturierten Produktautomaten liefert eine ähnlich Laufzeiteffiziente Spezifikation wie die der Bildung des Produktautomaten. Die aus diesen Methoden entstehenden Implementationen sind deutlich schneller wie die, die mit der herkömmlichen Methode, jeweils ein Automat pro Schicht, erzielt werden. Der Aufwand, der bei der Erstellung und Wartung der Spezifikation der Schichten anfällt, ist bei der Methode des strukturierten Produktautomaten in etwa genauso hoch wie bei der herkömmlichen Methode und damit bedeutend geringer als bei der Methode der Bildung des Produktautomaten.

Die Methode des strukturierten Produktautomaten kombiniert also die Vorteile der Produktautomaten-Methode, im Hinblick auf Effizienz der Implementierung, und der herkömmlichen Methode, im Hinblick auf Kosten bei der Erstellung und Wartung der Spezifikation.

Literatur

[Bre93] J. Bredereke : Eine Estelle-Erweiterung für strukturiertere Spezifikationen und für einen neuen Spezifikationsstil, interner Bericht, 1993

[BuDe87] S. Budkowski, P. Dembinski : An introduction to Estelle: a specification language for distributed systems, Comp. Networks and ISDN Systems (V.14 p. 3-23), 1987

[Got94] R. Gotzhein : Skript zur Vorlesung Spezifikation von Kommunikationssystemen, Universität Kaiserslautern, 1994

[Hof93] B. Hofmann : Integration von Darstellungs- und Session-Schicht in Estelle, Kommunikation in verteilten Systemen (Ed.: N. Gerner and H.-G. Hegering and J. Swoboda), GI/ITG-Fachtagung, Springer Munich, 1993

[ISO7498] ISO 7498 : Basis reference model for open systems interconnection, ISO/TC 97/SC 16, 1981

[ISO8322] ISO 8322 : Connection oriented presentation service definition, ISO/TC 97, 1988

[ISO8323] ISO 8323 : Connection oriented presentation protocol specification, ISO/TC 97, 1988

[ISO8326] ISO 8326 : Basic connection oriented session service definition, ISO/TC 97, 1987

[ISO8327] ISO 8327 : Basic connection oriented session protocol specification, ISO/TC 97, 1987

[ISO9074] Estelle : A formal description technique based on an extended state transition model, ISO/TC 97/SC 21, 1989

[Tan92] Andrew S. Tanenbaum : Computer-Netzwerke, Wolfram's Fachverlag, Attenkirchen, 1992

[Tur93] K. Turner : Using formal description techniques, Wiley , 1993

A Änderungen in Dateien

In allen Dateien wurden vorhandene writeln-Anweisungen entfernt. Eine entsprechende primitive Prozedur wird von pet/dingo nicht zur Verfügung gestellt, weshalb eine aufwendige Umarbeitung der Bildschirmausgaben notwendig gewesen wäre. Da diese Ausgaben aber größtenteils nur als Debug-Hilfen dienten, indem z.B. die Daten einer eingehenden Interaktion ausgegeben wurden, konnte darauf verzichtet werden. Nur in dem Modul shell.e dienten die Anweisungen zur Kommunikation mit dem Anwender. Der Anwender konnte auswählen, welches Dienstprimitiv der Darstellungs-Schicht als nächstes ausgeführt werden sollte. Für die durchzuführenden Messungen ist eine Kommunikation mit dem Anwender aber nicht erforderlich, sondern im Gegenteil würde sie die Meßergebnisse verfälschen.

type. e Die Datei type.e enthält Definitionen aller Paket-Datentypen, die die Schichten 5 und 6 erreichen. Neben den Paket-Typen der Sitzungs– und Darstellungsschicht sind dies noch die Paket-Typen der angrenzenden Schichten, der Anwendungs– und der Transportschicht. Außer diesen Typdefinitionen sind noch alle dafür benötigten Hilfsdatentypen und Konstanten definiert.

Bei der Konstanten MAX_CN wurde der zugewiesene Wert von 10240 auf 1024 vermindert, da mit dem alten Wert ein Aufteilen der Userdaten der SCONreq-Anforderung auf mehrere Steuerpakete der Sitzungsschicht nicht möglich war. Die Konstante MAX_CN gibt die maximale Anzahl an Datenoctets an, die in die Userdata-Komponente des CN-Steuerpaketes passen. Ist die Anzahl der Userdaten in der SCONreq-Anforderung größer als diese Konstante, so sieht das Protokoll vor diese Userdaten auf mehrere Steuerpakete (CN + CDOs) zu verteilen. Da der alte Wert von MAX_CN aber größer als MAX_DATA (maximale Anzahl der Userdaten in der SCONreq-Anweisung = 2048) war, trat dieser Fall nie ein. Die Steuerpakete konnten 5 mal soviel Userdaten aufnehmen, wie mit der Anforderung ankommen konnten. Durch den neuen Wert hat sich das Verhältnis MAX_CN : MAX_DATA auf 0.5 gesenkt, d. h. ein Steuerpaket kann nur noch die Hälfte der maximal möglichen zu übertragenden Userdaten aufnehmen. Damit ist gewährleistet, daß die Userdaten der SCONreq-Anforderung auf höchstens ein CN– und ein CDO–Steuerpaket aufgeteilt werden.

Der Datentyp DATA_STRING_TYPE wurde in der Original-Version nicht weiter spezifiziert. Um eine lauffähige Spezifikation zu erstellen, muß aber jeder Datentyp spezifiziert sein. DATA_STRING_TYPE wurde als ein Array von Charactern mit MAX_DATA Elementen definiert.

SESSION_REQUIREMENT_TYPE ist jetzt als ein Set von SESSION_REQUIREMENT_DEF definiert. Dies ist eine direktere Umsetzung des in der OSI-Norm vorgeschlagenen Datentypes nach Estelle, als dies mit einen Bit-Array über alle Komponenten des Aufzählungstyps SESSION_REQUIREMENT_DEF der Fall war.

Die beiden Datentypen SESSION_REFERENCE_TYPE und SESSION_ADDITIONAL_TYPE wurden verändert, weil sie in der ursprünglichen Implementation in einer C-Funktion von einem Pakettyp in einen anderen umgepackt wurden. Dabei wurden die beiden Datenstrukturen wie ein C–String behandelt, also die belegten Elemente mit einer 0-Markierung terminiert. Mit Hilfe der Standardfunktion strlen wurde die Anzahl der belegten Elemente des Strings bestimmt und nur diese Elemente umgepackt. In Pascal, das eine Untermenge von Estelle bildet, ist diese Definition eines Strings nicht üblich. Dort speichert man neben den einzelnen Elementen auch noch ihre Anzahl in einer Stringlängenkomponente ab. Da die Umpackfunktionen von C nach Estelle umgeschrieben werden sollten, wurden die Datentypen auch entsprechend der Sprachkonvention angepasst.

ROSE_ASN1_TYPE war nicht weiter spezifiziert. Da leider keine Information über den Aufbau dieses Datentyps in den verfügbaren ISO-Normen vorlag, wurde der Datentyp mit einem einfachen Standarddatentyp, dem Datentyp integer, gleichgesetzt. Der Datentyp wird nur zwischen der Darstellungsschicht und der Anwendungsschicht benutzt, um eintreffende bzw. abgehende Datenpakete mit den Dienstprimitiven DT_req und DT_ind auszutauschen. Als Parameter für diese Funktionen wurde ein Zeiger auf den Datentyp ROSE_ASN1_TYPE übergeben. Dies verbietet pet/dingo. In einer Interaktion dürfen keine Zeiger auf Daten zwischen zwei Automaten ausgetauscht werden. Deswegen mußte noch der ROSE_ASN1_POINTER_TYPE geändert werden. Er ist jetzt äquivalent zu ROSE_ASN1_TYPE, d. h. es wird immer die komplette Datenstruktur übergeben und nicht nur ein Zeiger auf sie.

channel. e Die Datei channel. e enthält alle Kanaldefinitionen, die in der Spezifikation benötigt werden.

Es wurde eine neue Kanaldefinition eingeführt, der Kanal Pres_Access_Channel_type. Er stellt die Vereinigung der zwei Kanäle Pres_Connection_Access_Channel_TYPE und Pres_Data_Access_Channel_TYPE dar. Diese beiden waren die Dienstzugangskanäle der Darstellungsschicht für den Verbindungsauf/abbau bzw. die Datenübertragung. Damit alle Module der Implementation die gleiche äußere Struktur besitzen, wurde die Teilung in zwei Dienstzugangskanäle entfernt und durch einen Kanal ersetzt.

session.e Die SPDUs der Sitzungs-Schicht werden als Datenpakete durch die Transport-Schicht gesendet. Prozeduren, die die SPDUs entprechend in Datenpakete verpacken und dabei auch die Benutzerdaten aus den eingehenden Interaktionen in diese Datenpakete umkopieren, waren als primitive Prozeduren deklariert und als C-Funktionen realisiert. Diese primitiven Prozeduren wurden nach Estelle transformiert. Anhand der Prozedur Encode_AA soll exemplarisch eine solche Transformation gezeigt werden.

In C war diese Funktion folgendermaßen definiert worden:

```
Encode_AA(machp, EnAA_tudata)
register struct SPCmachine *machp;
T_DATA_TYPE * EnAA_tudata;
{
    EnAA_tudata->Tdata[0] = 26;
    EnAA_tudata->Tdata[1] = 0;
    EnAA_tudata->Tlen = 2;
    EnAA_tudata->Tflag = DATA_END;
}
```

In Estelle wurde die Prozedur folgendermaßen definiert:

```
procedure Encode_AA(var EnAA_data : T_DATA_TYPE);
begin
    EnAA_data.Tdata[1] := chr(26);
    EnAA_data.Tdata[2] := chr(0);
    EnAA_data.Tlen := 2;
    EnAA_data.Tflag := DATA_END;
end;
```

Auf den ersten Parameter der C-Funktion konnte verzichtet werden, er wurde von der Laufzeitumgebung des anderen Compiler an die primitiven Prozeduren übergeben und hat für die eigentliche Aufgabe der Funktion keine Bedeutung. Anstelle eines Zeigers auf die Datenstruktur T_DATA_TYPE wie in der C-Funktion wurde in Estelle eine Referenz übergeben. Zu beachten sind noch die unterschiedlichen Indexwerte des ersten Elements eines Feldes. In C enthält das 0-te Feldelement das erste und in Estelle das 1-te das erste Element.

Alle anderen primitiven Prozeduren der Sitzungs-Schicht wurden nach dem gleichen Schema nach Estelle transformiert.

presentation. e Die Darstellungs-Schicht benötigt ebenfalls Prozeduren, die die Nutzerdaten aus den eingehenden Interaktionen in die ausgehenden Interaktionen umkopiert. Zusätzlich ist bei diesem Umkopieren noch die Konvertierung nach ASN.1-Darstellung vorgesehen. Auch diese Prozeduren waren wieder, wie in der Sitzungs-Schicht, als primitive Prozeduren deklariert und als C-Funktionen realisiert. Auf eine Transformation nach Estelle wurde verzichtet, sie ist analog zu Schicht 5 durchzuführen. Stattdessen wurden nur dummy-Prozeduren in Estelle definiert, die die Datenfelder der ausgehenden Interaktionen korrekt initialisieren, sonst aber nichts leisten. D.h. daß Benutzerdaten, welcher der Darstellungs-Schicht zum Versenden übergeben werden, von dieser weggeworfen werden und stattdessen ein Datenpaket der Länge Null verschickt wird. Dies reicht zur Durchführung der Messungen voll aus und garantiert zudem noch eine bei allen Datenpaketen gleichlange Übertragungszeit.

B Meßwerte

In den folgenden Tabellen sind die gemessenen Laufzeiten in Sekunden für die verschiedenen Kombinationen aus Abschnitt 7.2 aufgelistet. Es wurden Messungen durchgeführt für die Übertragung von 1, 10 und 100 Datenpakete-ten. Jede dieser Messungen erfolgte zehnmal, wobei der erste Meßwert nicht berücksichtigt wurde. Beim ersten Meßwert war der komplette Programmcode noch nicht im Cache des Prozessors vorhanden und mußte erst geladen werden. Die folgenden Messungen lieferten bedeutend kleinere Meßwerte als die erste, da bei ihnen der Cache schon entsprechend geladen war. Durch das Ignorieren des ersten Meßwertes konnte die Genauigkeit der Gesamtmessung erhöht werden. In den Tabellen sind außer den Meßwerten auch noch der Mittelwert ($\bar{x}$), die Varianz (σ^2), die Standardabweichung (σ), sowie die relative Standardabweichung ($\sigma/\bar{x}$) in Prozent für Messungen mit gleicher Anzahl Datenpakete angegeben.

	1 Datenpaket	10 Datenpakete	100 Datenpakete
	0.266656	0.416650	1.949922
	0.266656	0.416650	1.983254
	0.249990	0.433316	1.933256
	0.266656	0.433316	1.916590
	0.266656	0.399984	2.033252
	0.249990	0.416650	1.983254
	0.266656	0.433316	1.933256
	0.266656	0.399984	1.899924
	0.249990	0.433316	1.933256
$\bar{x}$	0.261101	0.420354	1.951774
σ^2	0.000069	0.000193	0.001697
σ	0.008333	0.013888	0.041199
$\sigma/\bar{x}$	3%	3%	2%

Tabelle 12: 1) Referenz - Referenz

	1 Datenpaket	10 Datenpakete	100 Datenpakete
	0.116662	0.216658	0.899964
	0.133328	0.216658	0.866632
	0.116662	0.216658	0.883298
	0.116662	0.199992	0.866632
	0.133328	0.199992	0.866632
	0.133328	0.199992	0.883298
	0.116662	0.199992	0.866632
	0.116662	0.199992	0.916630
	0.133328	0.199992	0.883298
$\bar{x}$	0.124069	0.205547	0.881446
σ^2	0.000077	0.000069	0.000309
σ	0.008784	0.008333	0.017568
$\sigma/\bar{x}$	7%	4%	2%

Tabelle 13: 2) Session-Integration - Session-Integration

	1 Datenpaket	10 Datenpakete	100 Datenpakete
	0.066664	0.099996	0.466648
	0.066664	0.116662	0.499980
	0.066664	0.083330	0.499980
	0.066664	0.099996	0.483314
	0.066664	0.099996	0.466648
	0.083330	0.099996	0.499980
	0.066664	0.099996	0.483314
	0.049998	0.099996	0.499980
	0.066664	0.099996	0.499980
$\bar{x}$	0.066664	0.099996	0.488869
σ^2	0.000069	0.000069	0.000208
σ	0.008333	0.008333	0.014433
$\sigma/\bar{x}$	12.5%	8%	3%

Tabelle 14: 3) strukt. Integration - strukt. Integration

	1 Datenpaket	10 Datenpakete	100 Datenpakete
	0.149994	0.233324	1.249950
	0.149994	0.249990	1.216618
	0.133328	0.283322	1.233284
	0.133328	0.249990	1.249950
	0.149994	0.266656	1.216618
	0.133328	0.233324	1.233284
	0.133328	0.249990	1.233284
	0.133328	0.249990	1.216618
	0.133328	0.249990	1.249950
$\bar{x}$	0.138883	0.251842	1.233284
σ^2	0.000069	0.000239	0.000208
σ	0.008333	0.015465	0.014433
$\sigma/\bar{x}$	6%	6%	1%

Tabelle 15: 4) Produkt-Integration - Referenz

	1 Datenpaket	10 Datenpakete	100 Datenpakete
	0.149994	0.249990	1.249950
	0.133328	0.249990	1.233284
	0.133328	0.249990	1.266616
	0.149994	0.266656	1.249950
	0.149994	0.233324	1.266616
	0.149994	0.249990	1.249950
	0.133328	0.233324	1.266616
	0.149994	0.249990	1.283282
	0.149994	0.283322	1.299948
$\bar{x}$	0.144439	0.251842	1.262912
σ^2	0.000069	0.000239	0.000401
σ	0.008333	0.015465	0.020030
$\sigma/\bar{x}$	6%	6%	2%

Tabelle 16: 5) strukt. Integration - Referenz