Untersuchung und Beschreibung unterschiedlicher Spezifikationsstile in der formalen Beschreibungstechnik Estelle anhand des „Association Control Service Elements“

von
Dirk Barthel

4. April 1996

Zusammenfassung

In dieser Projektarbeit werden 5 verschiedene Spezifikationsstile in der formalen Beschreibungstechnik Estelle auf Lesbarkeit, Einfluß der Stilrichtung auf die Laufzeiteffizienz und auf mögliche bzw. notwendige Estelle–Erweiterungen zur Realisierung der Stilrichtungen untersucht. Als Grundlage der Untersuchung dient dabei das „Association Control Element“. Die betrachteten Stile sind die beiden klassischen zustandsorientierten und nachrichtenorientierten Stile, der nachrichtenorientierte Stil ohne expliziten Hauptzustand, der nachrichtenorientierter Stil mit unifizierten Nachrichten und der prozeduraler Stil. Es hat sich herausgestellt, daß der klassische nachrichtenorientierte Stil in Bezug auf die Lesbarkeit die besten Ergebnisse liefert, und, daß mit dem prozeduralen Stil eine Steigerung der Laufzeiteffizienz um den Faktor 3,6 gegenüber den anderen Stilen möglich ist.

FB Informatik
AG Rechnernetze

Professor Dr. Reinhard Gotzhein

Aufgabe und Betreuung: Dipl.–Inform. Jan Bredereke

Erklärung

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

Kaiserslautern, den 4. April 1996

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Arbeit ist in den allgemeinen Rahmen zur Verbesserung der formalen Beschreibungstechnik Estelle im Hinblick auf Lesbarkeit und Laufzeiteffizienz einzuordnen, einem der Ziele der AG Rechnernetze des FB Informatik an der Universität Kaiserslautern.

Schwerpunkte dieser Arbeit sind die Untersuchung und Beschreibung verschiedener Spezifikationsstile in der Spezifikationssprache Estelle. Die verschiedenen Spezifikationsstile werden dabei auf ihre Lesbarkeit und Auswirkungen auf die Laufzeiteffizienz untersucht. Dazu kommt die Frage nach möglichen Estelle–Erweiterungen zur Verbesserung der Lesbarkeit bzw. nach notwendigen Erweiterungen zur Realisierung der einzelnen Stile.

Die in Estelle zu beschreibenden und zu untersuchenden Stilrichtungen sind:

Stil 1: Zustandsorientierter Stil

Stil 2: Nachrichtenorientierter Stil

Stil 3: Nachrichtenorientierter Stil ohne expliziten Hauptzustand

Stil 4: Nachrichtenorientierter Stil mit unifizierten Nachrichten

Stil 5: Prozeduraler Stil

Der zustandsorientierte und der nachrichtenorientierte Stil sind die klassischen Stilrichtungen in Estelle. Sie unterscheiden sich durch die Sichtweise auf das zu spezifizierende Problem, normalerweise ein erweiterter, endlicher Protokollautomat. Je nach Sichtweise auf den Protokollautomaten stehen die Zustände oder die Nachrichten im Vordergrund. Dies läßt sich in der Spezifikationssprache Estelle durch Anordnen bestimmter Klauseln der Estelle–Transitionen erreichen.

Der nachrichtenorientierte Stil ohne expliziten Hauptzustand entsteht aus dem normalen nachrichtenorientierten Stil durch Verschiebung der Kontrollzustände in die Transitionsblöcke, d.h. es werden die „From"–Klauseln in dieser Stilrichtung nicht mehr benötigt.

Die Stilrichtung nachrichtenorientiert mit unifizierten Nachrichten ist ebenfalls aus der klassischen nachrichtenorientierten Stilrichtung abgeleitet. Der Unterschied zu dem dritten Stil besteht darin, daß die Nachrichten in variante Records zusammengefaßt werden.

Der prozedurale Stil geht schließlich auf einen Vorschlag von J. Brederek (siehe [1]) zurück. In dieser Stilrichtung wird versucht, alle Estelle–Transitionen in Prozeduren zu wandeln, um dadurch die laufzeitineffiziente Kommunikation zwischen Estelle–Modulen durch effizientere Prozeduraufrufe ersetzen zu können. Diese Ersetzung wird als Verschmelzung von Estelle–Modulen bezeichnet. Den nachrichtenorientierte Stil ohne expliziten Hauptzustand kann man dabei als Zwischenschritt bei der Umsetzung in den prozeduralen Stil ansehen.

Diese verschiedenen Spezifikationsstile in der Spezifikationssprache Estelle werden in Kapitel 3 auf die Lesbarkeit der einzelnen Stile und Verbesserungen der Lesbarkeit durch Erweiterungen von Estelle untersucht.

In Kapitel 4 steht dagegen die Frage der Laufzeiteffizienz im Vordergrund. Dabei interessiert speziell die Frage, welchen Einfluß die Stilrichtung auf die Laufzeiteffizienz hat, und besonders, welcher Vorteil der prozedurale Stil gegenüber den anderen Stilrichtungen in Bezug auf die Laufzeiteffizienz besitzt.

Als Grundlage für die Untersuchungen auf Lesbarkeit und Laufzeiteffizienz dient das „Association Control Service Element", im Folgenden auch mit ACSE abgekürzt. Der ACSE–Dienst ist auf Schicht 7 des OSI–Basisreferenzmodells angesiedelt und stellt als Dienstleistungen den Aufbau, Abbau und Abruch von Assoziationen zur Verfügung. Associationen stellen im Kontext der Schicht 7 Beziehungen zwischen Nutzern des ACSE–Dienstes und Verbindungen der Darstellungsschicht (Schicht 6) dar. Das Kapitel 2 behandelt den ACSE–Dienst und den zur Realisierung des Dienstes notwendigen ACSE–Protokollautomaten. Außerdem versucht das Kapitel 2 die Schwächen des ACSE–Protokollautomaten, gerade beim Abbau von Assoziationen, aufzuzeigen.

Zum Abschluß dieser Einleitung noch einige Bemerkungen zu der von ISO standardisierte Spezifikationssprache Estelle. Estelle ist eine formale Beschreibungstechnik, entwickelt für die Beschreibung verteilter, informationsverarbeitender Systeme, speziell für die Beschreibung von Protokollen des OSI–Basisreferenzmodells. Dabei bedeutet formal, daß der Spezifikationssprache Estelle eine formale Semantik zu Grunde liegt, die zum Beispiel die Ausführungsreihenfolge von Estelle–Transitionen bestimmt. Dadurch ist es möglich, das gewünschte Verhalten einer Spezifikation nachzuweisen.

Eine der wesentlichen Eigenschaften von Estelle ist es, die Zustandsübergänge eines erweiterten, endlichen Automaten durch Estelle–Transitionen zu beschreiben. Eine solche Transition besteht aus Schaltbedingungen und Schaltwirkungen. Zu den Schaltbedingungen einer Estelle–Transition zählen zum Beispiel die Klauseln für die Zustände und Nachrichten. Die Zustände werden in einer Transition durch die „From"–Klauseln, die zu empfangenden Nachrichten durch die „When"–Klausel angegeben. Als Schaltwirkungen sind dagegen Angaben, die den Folgezustand ("To"–Klausel) nach dem Feuern einer Transition kennzeichnen und der auszuführende Transitionsblock, hier besonders die "Output"–Anweisungen zu nennen.

Je nach Anordnung der Schaltbedingungen läßt sich eine nachrichten– oder zustandsorientierte Stilrichtung in einer Spezifikation erreichen. Die formale Semantik von Estelle legt dabei fest, daß durch unterschiedliche Anordnugen von Schaltbedingung das Verhalten der Spezifikation sich nicht ändert.

Zur Verbesserung der Lesbarkeit einer in Estelle geschriebenen Spezifikation, bietet diese Sprache zum Beispiel die Möglichkeit, mehrere einfache Transitionen zu geschachtelten Transitionen zusammen zu fassen. Von dieser Möglichkeit wird in der Beschreibung des ACSE–Protokolles mittels Estelle intensiv gebrauch gemacht. In Abschnitt 3.3 bzw. in den Estelle–Quelltexten für den ACSE–Protokollautomaten sind Beispiele für geschachtelte Transitionen zu finden.

Für eine genauere Beschreibung der Spezifikationssprache Estelle sei auf den Standard [8], die Bücher [5] und [12] verwiesen. Eine gute und kurze Übersicht bieten die Folienkopien zur Vorlesung „Spezifikation von Kommunikationssystemen" von Prof. Dr. R. Gotzhein (siehe [4]).

2 Das Association Control Service Element (ACSE)

Das „Association Control Service Element“ (im Folgenden mit „ACSE“ abgekürzt) ist auf der Schicht 7 des OSI–Basisreferenzmodells angesiedelt.

Der Dienst stellt eine Reihe von Dienstleistungen zur Kontrolle einer Assoziation zur Verfügung. Eine Assoziation stellt im Kontext der Schicht 7 eine Beziehung zwischen Dienstnutzern und Verbindungen der Darstellungsschicht (Schicht 6) dar. Es besteht eine 1:1 Beziehung zwischen Assoziation und Verbindung. Eine Assoziation ist deshalb eine Ende–Zu–Ende Verbindung der Darstellungsschicht.

Dienstleistungen des ACSE–Dienstes sind Aufbau, Abbau und Abbruch einer Assoziation bzw. Verbindung zwischen kommunizierenden Dienstnutzern. Aufbau und Abbau sind bestätigte, Abbruch von Assoziationen sind unbestätigte Dienstleistungen des ACSE–Dienstes. Das bedeutet, daß die Dienstnutzer einem Verbindungsauf– bzw. einem Verbindungsabbau zustimmen müssen, bevor dieser tatsächlich durchgeführt wird, während für einen Abbruch der Verbindung keine Zustimmung erforderlich ist. Im Folgenden wird mit dem Abbruch einer Verbindung ein möglicher Informationsverlust und mit dem Abbau ein Auflösen der Verbindung ohne Informationsverlust bezeichnet.

Der ACSE–Protokollautomat baut auf dem Dienst der darunterliegenden Darstellungsschicht des OSI–Basisreferenzmodells auf und beschreibt ein symmetrisches Protokoll, d.h. beide an einer Verbindung teilnehmende Automaten besitzen die gleiche Funktionalität. Die Funktionalität des ACSE–Protokollautomaten kann in drei bzw. je nach Spezifikationsstil oder Implementation in vier Teile untergliedert werden:

1. Aufbau
2. Abbau (ohne Informationsverlust)
3. Abbruch (mit möglichem Informationsverlust)
4. (Fehlerbehandlung)

Der vierte Teil ist für die Behandlung von möglichen Empfangsfehlern zuständig.

Für das bessere Verständnis des ACSE–Protokolls ist in Abbildung 1 eine typische Anordnung der einzelnen Protokollinstanzen innerhalb des OSI–Basisreferenzmodells und ihre Verbindungen untereinander dargestellt. Es bezeichnen „User_A“ und „User_B“ die jeweiligen Dienstnutzer der ACSE–Protokollinstanzen, „ACSE_A“, „ACSE_B“ die jeweiligen Protokollinstanzen für die Dienstnutzer und „PRES_A“, „PRES_B“ die entsprechenden Protokollinstanzen der Darstellungsschicht. Darunter sind die restlichen Schichten des OSI–Basisreferenzmodells angesiedelt. Prinzipiell bilden alle mit dem Buchstaben „A“ bzw. „B“ endenden Protokollinstanzen eine für die Kommunikation notwendige logische Einheit.

Als Ausgangspunkt für die weiteren Betrachtungen sind für eine Verbindung die Instanzen mit der Endung „A“ gewählt. Alle für diese Einheit gemachten Aussagen gelten in gleicherweise für die andere Einheit, falls diese als Ausgangspunkt gewählt wird.

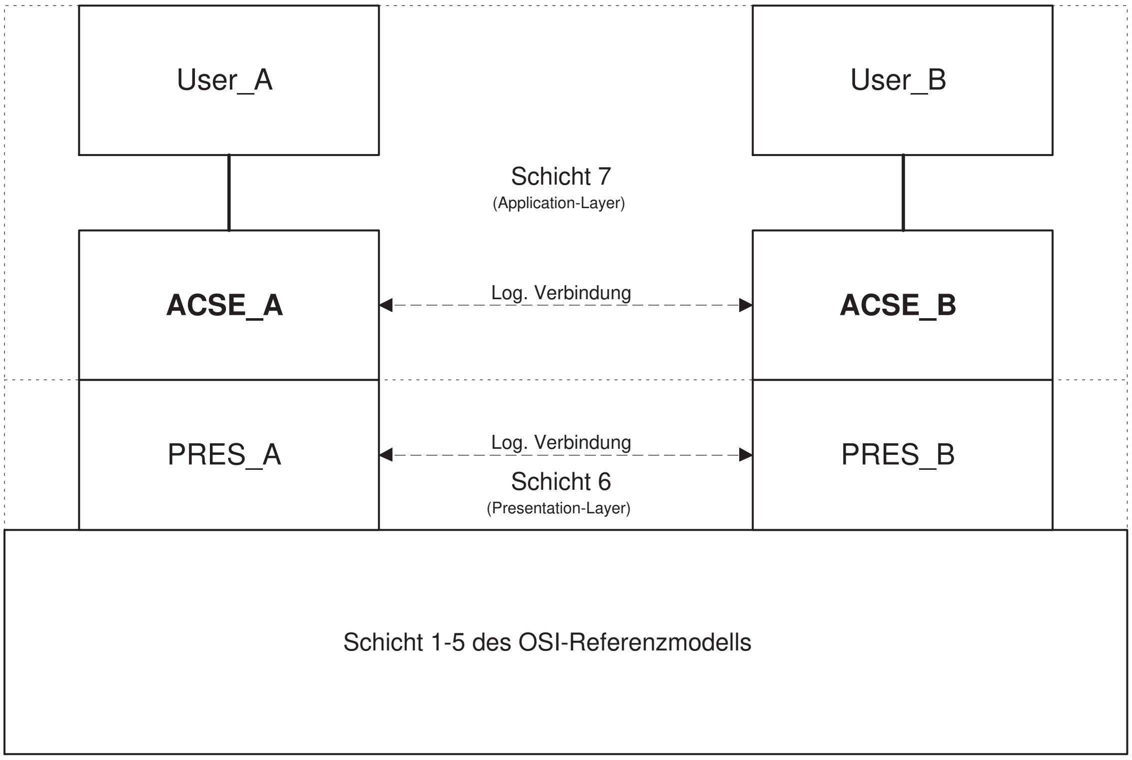

Abbildung 1: Typische Anordnung für die Protokollinstanzen

In den folgenden Abschnitten dieses Kapitels wird auf die entsprechenden Teile und ihre Funktion innerhalb des ACSE–Protokolles näher eingegangen. Zuvor ein paar Bemerkungen über die Bezeichnungen von Nachrichten und Zuständen.

2.1 Namenskonventionen

Für die Nachrichten, Zustände und Typen sind, bis auf einige kleine Änderungen, die Bezeichnungen des zur Verfügung gestellten Originalquelltextes übernommen worden. Die Bezeichnungsänderungen bei einigen Nachrichten und den dazugehörigen Typen für die Nachrichtenparameter haben zwar den Nachteil einer Änderung der Kanaldefinition für den Kanal zwischen den Dienstnutzern und den ACSE–Protokollinstanzen, bewirken aber eine Verbesserung der Lesbarkeit durch eine einheitlichere Bezeichnungskonvention.

Ein Beispiel für die Änderung einer Bezeichnung ist das Umbenennen von „A_ASSRQ“ in „A_ASSreq“ und „A_ASSRQ_TYPE“ in „A_ASSreq_TYPE“ für die Nachricht bzw. für die Typen der Nachrichtenparameter[1].

[1] Es sei darauf hingewiesen, daß die Änderungen das Verhalten des Dienstes nicht beeinflussen !

Die Namenskonvention für die Nachrichten entspricht folgendem, einfachem Aufbau:

- **A_***
  Bezeichnet Nachrichten, die ein Nutzer des Dienstes an einen ACSE–Protokollautomaten schicken bzw. von diesem empfangen kann.
- **P_***
  Dies sind Nachrichten, die von der Darstellungsschicht („Presentation–Layer“) empfangen oder an diese gesendet werden können.

Entsprechend der drei Dienstleistungen des ACSE–Protokolls können die Nachrichten weiter unterteilt werden:

- ***_ASS*** (Association)
  _CON (Connection)
  Kennzeichnet alle am Verbindungsaufbau beteiligten Nachrichten.
- ***_REL*** (Release)
  Dies sind alle am Verbindungsabbau beteiligten Nachrichten.
- ***_AB*** (Abort)
  _UAB (User–Abort)
  _PAB (Provider–Abort)
  Bezeichnet alle für den Abbruch einer Verbindung verantwortlichen Nachrichten.

„*_ASS*“ steht für Dienstprimitive des ACSE–Dienstes und „*_CON*“ für Dienstprimitive der Darstellungsschicht.

Welche Nachrichten eine Protokollinstanz allgemein empfangen oder senden kann, untergliedert sich in

- ***req** (Request)
  ***rsp** (Response)
  für die zu empfangenden und
- ***ind** (Indication)
  ***cnf** (Confirm)
  für die zu sendenden Nachrichten.

Es bedeutet zum Beispiel „A_ASSreq“, daß diese Nachricht zur Schnittstelle des ACSE–Dienstes gehört („A_“), daß die Nachricht zur Gruppe des Verbindungsaufbaus zu zählen ist („_ASS“) und daß es sich um eine Nachricht handelt, die von dem ACSE–Protokoll nur empfangen werden kann („req“).

Für die 8 Zustände des Protokollautomaten gilt folgende Konvention:

- „**IDLE**" und „**ASSOCIATED**" bezeichnen die jeweiligen Zustände, in denen eine Verbindung existiert (ASSOCIATED) bzw. nicht existiert (IDLE)
- „**AWAIT_***" sind benötigte Zwischenzustände, in denen auf eine Bestätigung gewartet wird.
- „**COLLISION_***" bezeichnet Zustände, die zur Behandlung möglicher Kollisionen beim Verbindungsabbau benötigt werden.

Die Zwischenzustände („AWAIT_*") teilen sich in

- „**AARE**" und „**ASSRP**" für den Verbindungsaufbau und
- „**RLRE**" und „**RLRP**" für den Verbindungsabbau auf.

Für die „COLLISION_*"–Zustände ist eine weitere Unterteilung in

- „**INITIATOR**" für den Initiator einer Verbindung und
- „**RESPONDER**" für den Akzeptor eines Verbindungswunsches

gegeben.

Weiterhin bezeichnet „**ABTSTATE**"[2] eine Zustandsmenge, in der bis auf „IDLE" alle anderen Zustände zusammengefaßt sind.

Die entsprechenden Bezeichnungen der Norm sind zusammen mit allen Nachrichten, Zuständen und ihren Bedeutungen im Anhang A nochmals aufgeführt.

2.2 Verbindungsaufbau

Der Teil mit dem Verbindungsaufbau ist im ACSE–Protokollautomaten gekennzeichnet durch die Zustände

1. **IDLE**
   Keine Verbindung ist aufgebaut, der Automat befindet sich im Ruhezustand.
2. **AWAIT_AARE**
   In diesem Zustand erwartet ACSE_A (Initiator der Verbindung) eine Nachricht von seinem Gegenüber (ACSE_B: Responder in der Verbindung). In der Nachricht teilt dieser die Entscheidung seines Dienstnutzers über den Aufbauwunsch mit.

[2] „ABT" steht dabei für „Abort"

3. **AWAIT_ASSRP**
   Zustand, in dem der Protokollautomat ACSE_B auf die Entscheidung seines Nutzers über einen Verbindungsaufbauwunsch von ACSE_A wartet.

4. **ASSOCIATED**
   Die Verbindung ist aufgebaut, die Datenübertragung kann stattfinden.

und die Nachrichten:

1. **A_ASSreq** (ACSE Association Request)
   Mit dieser Nachricht äußert User_A einen Aufbauwunsch.

2. **A_ASSind** (ACSE Association Indication)
   Der Wunsch nach einer Verbindung wird bei User_B mit dieser Nachricht angezeigt.

3. **A_ASSrsp** (ACSE Association Response)
   User_B akzeptiert oder lehnt damit den Aufbauwunsch von User_A ab.

4. **A_ASScnf** (ACSE Association Confirm)
   Mit dieser Nachricht zeigt ACSE_A die Bestätigung oder aber Ablehnung der Verbindung bei User_A an.

5. **P_CONreq** (PRES Connection Request)
   Damit fordert ACSE_A die Darstellungsschicht auf, eine Verbindung zu etablieren.

6. **P_CONind** (PRES Connection Indication)
   Die Darstellungsschicht zeigt hiermit ACSE_B an, daß ein Verbindungsaufbauwunsch vorhanden ist.

7. **P_CONrsp** (PRES Connection Response)
   Mit dieser Nachricht teilt ACSE_B die Entscheidung des Dienstnutzers User_B bzgl. des Aufbauwunsches von User_A der Darstellungsschicht (PRES_B) mit.

8. **P_CONcnf** (PRES Connection Confirm)
   Die Darstellungsschicht PRES_A teilt hiermit ACSE_A die Entscheidung des User_B's mit.

9. **A_PABind** (ACSE Provider Abort Indication)
   ACSE_A zeigt hiermit dem aufrufenden Dienstnutzer (User_A) an, daß es einen Verbindungsaufbau nicht unterstützen kann. (Parameter der A_ASSreq Nachricht sind nicht korrekt gesetzt. Der Protokollautomat arbeitet z.B. in einem anderen Modus)

Für die Entscheidung, wann eine Verbindung zustande kommt, sind die Parameter der einzelnen Nachrichten verantwortlich:

- **Mode** (als Parameter von A_ASSreq)
  - Werte:
    MODE_X410_1984 und MODE_NORMAL
  - Damit wird festgelegt, in welchem Modus das ACSE–Protokoll betrieben wird. Im Weiteren wird der Modus MODE_NORMAL besprochen. Für Informationen zu dem anderen Modus sei auf die entsprechende Norm (siehe [2]) verwiesen.
- **User_Data.Protocol_Version** (als Parameter von P_CONind)
  - Werte:
    VERSION_NONE und VERSION_ONE
  - Stimmt die Protokoll Version der beiden ACSE–Protokollinstanzen nicht überein, dann wird ein Verbindungsaufbau abgelehnt.[3]
- **Result** (als Parameter von A_ASSrsp)
  - Werte:
    ACCEPTED, RESP_REJECT_TRANSIENT und RESP_REJECT_PERMANENT
  - Ist der Wert des Parameters ACCEPTED, dann hat der entsprechende andere Teilnehmer der Verbindung (User_B) den Verbindungswunsch akzeptiert.
- **Result** (als Parameter von P_CONcnf)
  - Werte:
    P_ACCEPTANCE, USER_REJECTION, PROVIDER_REJECT_TRANSIENT und PROVIDER_REJECT_PERMANENT
  - Mit P_ACCEPTANCE teilt die Protokollinstanz (PRES_A) dem Protokollautomaten ACSE_A mit, daß der Verbindungswunsch von User_B akzeptiert worden ist.

Damit ergibt sich der in Abbildung 2 gezeigte Zustandsübergangsgraph.

Die Nachricht, die vor dem „/“ steht, ist diejenige, die empfangen werden muß, um von einem Zustand in den entsprechenden anderen Zustand zu wechseln. Die Nachricht, die hinter dem „/“ steht, gibt an, welche Folgenachricht beim Zustandswechsel gesendet wird.

Ein „+“ bzw. „–“ in den Klammern kennzeichnet Nachrichten, deren Parameter den Aufbau einer Verbindung beeinflussen. Diese Ergebnisparameter beinhalten die Entscheidungen der Dienstnutzer über einen Verbindungswunsch. „Positive“ Parameter bewirken einen Verbindungsaufbau, während „negative“ Parameter eine Verbindung nicht zustande kommen lassen.

[3] Die Bedeutung des Parameters ist Sache der jeweiligen Implementierung („Matter of Implementation“).

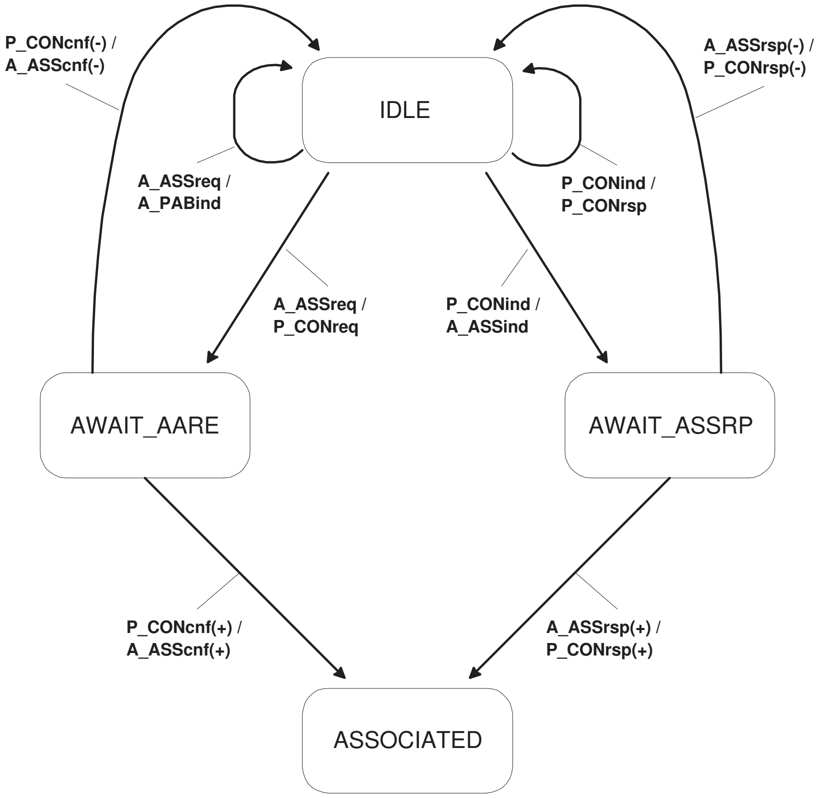

Abbildung 2: Zustandsübergangsgraph für den Verbindungsaufbau

Ein erfolgreicher Verbindungsaufbau findet aus Sicht der ACSE–Protokollautomaten in zwei Phasen[4] statt:

1.Phase:

Beide Automaten befinden sich im Ruhezustand IDLE. Es besteht keine Verbindung.

Der Protokollautomat ACSE_A empfängt ein A_ASSreq(+) und sendet ein P_CONreq(+) an PRES_A. Der Automat ACSE_A wechselt in den Zustand AWAIT_AARE.

ACSE_B erhält eine P_CONind(+) Nachricht und gibt mit einem A_ASSind(+) den Aufbauwunsch von User_A an seinen Nutzer, User_B, weiter. Der Automat wechselt in den Zustand AWAIT_ASSRP. In diesem Zustand wartet der Automat auf ein A_ASSrsp von User_B.

[4]Die Phasen sind entsprechend der Synchronisationspunkte der beiden Automaten eingeteilt.

2. Phase:

ACSE_B empfängt von seinem Dienstnutzer ein A_ASSrsp(+). Daraufhin gibt ACSE_B die Antwort als P_CONrsp(+) an PRES_B weiter und wechselt in den Zustand ASSOCIATED.

ACSE_A erhält von PRES_A die Bestätigung des Aufbauwunsches durch ein P_CONcnf(+). ACSE_A gibt ein A_ASScnf(+) an seinen Dienstnutzer weiter und wechselt in den Zustand ASSOCIATED.

Beide Automaten befinden sich im Zustand ASSOCIATED. Die Verbindung ist etabliert.

In Abbildung 3 ist das Zeitdiagramm („Time–Sequence–Diagram“) für einen erfolgreichen Verbindungsaufbau dargestellt.

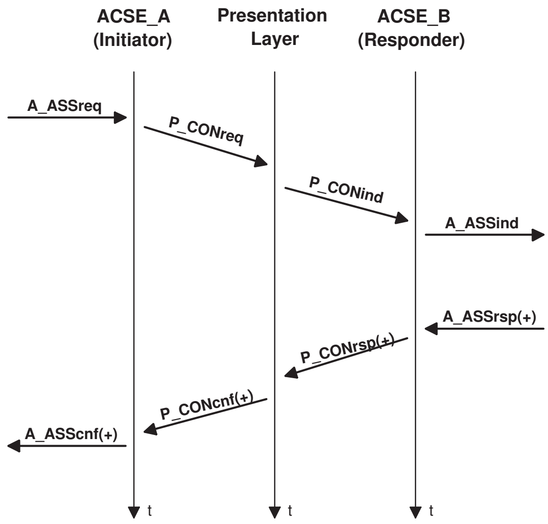

Abbildung 3: Zeitdiagramm für erfolgreichen Verbindungsaufbau

Als Konvention für den Aufbau von Verbindungen gilt, daß keine, eine oder zwei eigenständige Verbindungen initiiert werden, wenn die beiden Dienstnutzer gleichzeitig einen Aufbauwunsch äußern. Der in dieser Arbeit spezifizierte Protokollautomat benutzt die Konvention, daß keine Verbindung initiiert wird. Der Automat behandelt das Eintreffen eines Aufbauwunsches im Zustand AWAIT_AARE als Empfangsfehler und wechselt in den Zustand IDLE mit einer entsprechenden Nachricht an den anderen Partner.

Für die beiden anderen Konventionen ist es notwendig, daß eine Protokollinstanz existiert, die einen gleichzeitigen Aufbauwunsch, zum Beispiel durch Erzeugung von zusätzlichen ACSE–Protokollinstanzen und Verteilung der Nachrichten auf diese Instanzen, behandelt.

2.3 Verbindungsabbau

Der Verbindungsabbau ist gekennzeichnet durch die Zustände

1. **ASSOCIATED**
   Gleiche Bedeutung wie beim Verbindungsaufbau, d.h. eine Verbindung ist etabliert.

2. **AWAIT_RLRE**
   In diesem Zustand erwartet der Protokollautomat eine Entscheidung über den Abbauwunsch.

3. **AWAIT_RLRP**
   Der Automat hat in diesem Zustand einen Abbauwunsch empfangen und wartet nun auf eine Entscheidung seines Dienstnutzers über die Annahme bzw. Ablehnung des Abbauwunsches.

4. **IDLE**
   Ruhezustand, in dem sich der Automat nach Abbau der Verbindung befindet.

5. **COLLISION_INITIATOR**
   Der Zustand dient zum Auflösen einer Kollision[5]. Dabei geht nur der Automat (ACSE_A), der die Verbindung initiiert hat, in diesen Zustand.

6. **COLLISION_RESPONDER**
   Dieser Zustand dient ebenfalls zur Behandlung von Kollisionen. Diesmal geht der beteiligte Partner einer Verbindung (ACSE_B bzw. der Responder) in diesen Zustand.

Den Verbindungsabbau kennzeichnenden Nachrichten sind

1. **A_RELreq** (ACSE Release Request)
   Mit dieser Nachricht äußert einer der Teilnehmer einen Abbauwunsch (zum Beispiel User_A).

2. **A_RELind** (ACSE Release Indication)
   Ein Abbauwunsch von User_A wird hiermit bei User_B signalisiert.

3. **A_RELrsp** (ACSE Release Response)
   Diese Nachricht gibt User_B an ACSE_B weiter. User_B teilt damit seine Entscheidung über einen Abbau der Verbindung mit.

4. **A_RELcnf** (ACSE Release Confirm)
   Die Nachricht dient dazu, User_A die Entscheidung von User_B mitzuteilen. Sie wird von ACSE_A an User_A weitergereicht.

[5]Eine Kollision entsteht dadurch, daß beide an einer Verbindung beteiligten ACSE–Protokollinstanzen gleichzeitig eine A_RELreq Nachricht von ihrem jeweiligen Nutzer empfangen haben.

5. **P_RELreq** (PRES Release Request)
   Der Abbauwunsch von User_A wird mit dieser Nachricht von ACSE_A an PRES_A weitergeleitet.

6. **P_RELind** (PRES Release Indication)
   PRES_B teilt in dieser Nachricht den Abbauwunsch von User_A an ACSE_B mit.

7. **P_RELrsp** (PRES Release Response)
   Die Entscheidung über einen Verbindungsabbau wird mit dieser Nachricht von ACSE_B an PRES_B weitergegeben.

8. **P_RELcnf** (PRES Release Confirm)
   Die Bestätigung oder aber Ablehnung eines Verbindungsabbaus wird von PRES_A durch diese Nachricht an ACSE_A weitergereicht.

Die für einen Verbindungsabbau wichtigen Parameter und Variablen sind

- **Result** (als Parameter von A_RELrsp)
  - Werte: ACSE_AFFIRMATIVE und ACSE_NEGATIVE
  - Mit ACSE_AFFIRMATIVE zeigt der Teilnehmer seine Zustimmung zu einem Verbindungsabbau an.
- **Result** (als Parameter von P_RELcnf)
  - Werte: P_AFFIRMATIVE und P_NEGATIVE
  - P_AFFIRMATIVE zeigt die Zustimmung zum Verbindungsabbau an.
- **Is_Initiator**
  - Boolean–Variable, deren initialer Zustand FALSE ist.
  - Diese Variable dient zum Auflösen einer Kollision. Sie wird beim Aufbau einer Verbindung durch denjenigen ACSE–Automaten auf TRUE gesetzt, der die Verbindung erfolgreich initiiert hat und beim Abbau wieder auf FALSE zurückgesetzt.

Es ergibt sich der in Abbildung 4 gezeigte Zustandsübergangsgraph für den Abbau einer Verbindung.

Der zeitliche Ablauf für einen erfolgreichen Verbindungsabbau entspricht prinzipiell dem zeitlichen Ablauf eines erfolgreichen Verbindungsaufbaus (siehe Abb. 3). Für *_ASS* und *_CON* ist dabei *_REL* einzusetzen.

Interessanter ist das Verhalten des ACSE–Automaten bei einer Kollision. Eine Kollision entsteht, wenn beide an einer Verbindung beteiligten Automaten eine A_RELreq–Nachricht empfangen, bevor die P_RELind Nachricht ihres jeweiligen Gegenübers sie erreicht hat.

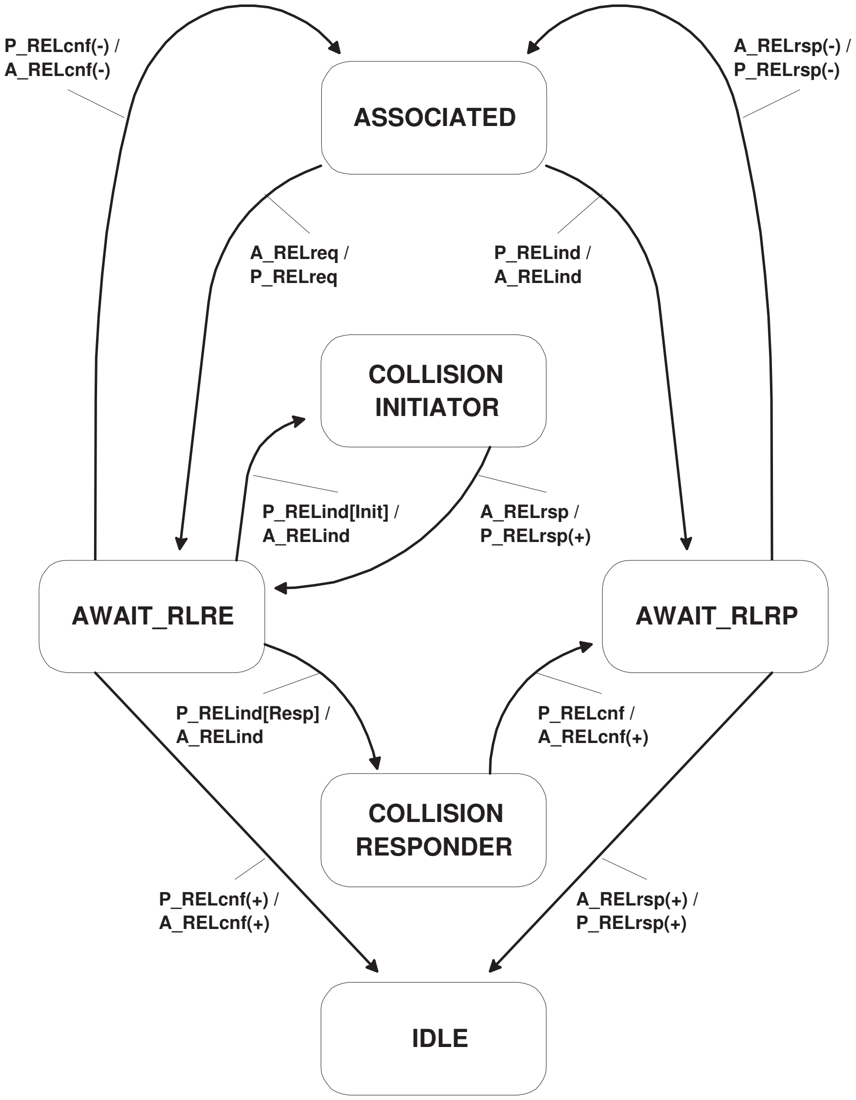

Abbildung 4: Zustandsübergangsgraph für Verbindungsabbau

Der Verbindungsabbau bei einer Kollision gestaltet sich in vier Phasen[6]:

1.Phase:

Beide Automaten befinden sich im Zustand ASSOCIATED. Die Verbindung ist etabliert.

ACSE_A empfängt ein A_RELreq und sendet ein P_RELreq an PRES_A. Der Protokollautomat wechselt von ASSOCIATED in den Zustand AWAIT_RLRE.

ACSE_B empfängt ein A_RELreq und sendet ein P_RELreq an PRES_B. ACSE_B wechselt damit auch in den Zustand AWAIT_RLRE.

[6]Die Phasen sind entsprechend der Synchronisationspunkte der beiden Automaten eingeteilt.

2. Phase: Kollisionserkennung

Mit Empfang der P_RELind Nachricht von PRES_A reicht der Automat an seinen Nutzer ein A_RELind weiter und wechselt in den Zustand COLLISION_INITIATOR[7]. In dem Zustand wartet der Automat auf eine A_RELrsp Nachricht von User_A.

ACSE_B empfängt ein P_RELind, sendet ein A_RELind an User_B und wechselt in den Zustand COLLISION_RESPONDER. In diesem Zustand wartet der Automat auf eine Bestätigung durch ein P_RELcnf(+).

3. Phase: Kollisionsbehandlung

Sobald ACSE_A ein A_RELrsp von User_A erhält, gibt es ein P_RELrsp(+)[8]an PRES_A weiter und wechselt wieder in den Zustand AWAIT_RLRE.

ACSE_B erhält ein P_RELcnf(+) als Bestätigung, sendet ein A_RELcnf(+) an User_B und wechselt in den Zustand AWAIT_RLRP. Damit ist der Abbauwunsch von User_B bearbeitet. Die Verbindung ist weiterhin etabliert!

4. Phase:

ACSE_B reicht die Bestätigung von User_B als P_RELrsp an PRES_B weiter und geht in den Ruhezustand IDLE über.

ACSE_A erhält von PRES_A die Bestätigung als P_RELcnf, wechselt nach IDLE und sendet ein A_RELcnf an User_A.

Beide Automaten befinden sich wieder im Ruhezustand. Die Verbindung ist damit aufgelöst.

Eine Kollision können die Dienstnutzer des ACSE–Dienstes durch den Empfang eines A_RELind nach dem Senden einer A_RELreq Nachricht erkennen.

In Abbildung 5 ist das Zeitdiagramm für eine Kollision dargestellt.

Für den Initiator und Responder in einer Verbindung hat eine Kollision die Konsequenz, daß der Dienstnutzer (User_A) des Initiators (ACSE_A) als nächstes eine A_RELrsp Nachricht senden muß, um eine Kollision zu beheben, während der Dienstnutzer (User_B) des Responders (ACSE_B) seine A_RELrsp Nachricht bis zum Eintreffen der Bestätigung seines Abbauwunsches verzögern muß (siehe ISO–Norm 8649, [6], Seite 9).

Würde der Dienstnutzer des Responders die A_RELrsp Nachricht bis zum Empfang der Bestätigung über seinen Abbauwunsch durch eine A_RELcnf Nachricht nicht verzögern, dann könnte

[7]Es wird angenommen, daß Is_Initiator für ACSE_A auf TRUE und für ACSE_B entsprechend auf FALSE gesetzt ist.

[8]ACSE_A sendet immer eine positive Entscheidung über den Abbauwunsch von User_B an PRES_A !

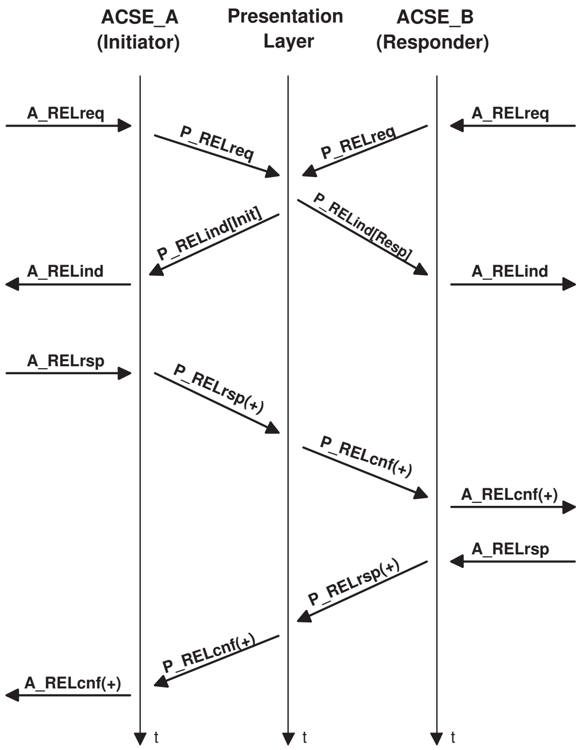

Abbildung 5: Zeitdiagramm für eine Kollision

die A_RELrsp Nachricht im Zustand COLLISION_ASSOC_RESPONDER vor einer P_RELcnf Nachricht eintreffen. Dies wird, entsprechend der ISO–Normen, in den vorliegenden Spezifikationen als Empfangsfehler behandelt.

Prinzipiell wäre es möglich, da die Spezifikationssprache Estelle asynchrone Verbindungen nutzt, die Annahme der A_RELrsp Nachricht bis zum Erreichen des entsprechenden Zustandes zu verzögern. Es müßte dazu lediglich der Zustand COLLISION_ASSOC_RESPONDER aus dem Fehlerbehandlungsteil für die Nachricht A_RELrsp genommen werden. Dies hätte aber zur Folge, daß für eine Implementation asynchrone Verbindungen erforderlich wären.

Da beide Nutzer einem Abbau zugestimmt haben, wäre es besser, direkt aus den Kollisionszuständen in den Zustand IDLE zu wechseln, ohne daß die Nutzer von einer Kollision etwas erfahren würden. Damit würde der Empfang einer A_RELcnf Nachricht beim Responder tatsächlich einen Verbindungsabbau bestätigen und das Verzögern der A_RELrsp Nachricht beim Dienstnutzer des Responders könnte entfallen. Die Dienstnutzer müßten den Sonderfall einer Kollision nicht mehr behandeln.

2.4 Abbruch einer Verbindung

Ein Verbindungsabbruch mit einem möglichen Verlust an Informationen kann, außer im Ruhezustand, in jedem anderen Zustand des ACSE–Automaten erfolgen. Der Verbindungsabbruch kann von einem Dienstnutzer des Protokollautomaten oder von dem darunterliegenden Diensterbringer ausgelöst werden.

Die für den Verbindungsabbruch relevanten Nachrichten sind

1. **A_ABreq** (ACSE Abort Request)
   Hiermit äußert der Dienstnutzer (z.B. User_A) den Abbruch der Verbindung

2. **A_ABind** (ACSE Abort Indication)
   Diese Nachricht zeigt dem anderen Teilnehmer (User_B) den Abbruch der Verbindung an. Dabei gilt, daß der Abbruch durch einen Nutzer der Verbindung ausgelöst worden ist.

3. **A_PABind** (ACSE Provider Abort Indication)
   Ausgelöst wird diese Nachricht durch den ACSE–Protokollautomaten oder durch die darunterliegenden Schichten. D.h. der Abbruch der Verbindung wurde durch die Diensterbringer („Provider“) verursacht.

4. **P_UABreq** (PRES User Abort Request)
   Mit dieser Nachricht teilt die ACSE–Protokollinstanz der PRES–Protokollinstanz mit, daß die Verbindung abgebrochen werden soll. Der Abbruch ist von einem Dienstnutzer („User“) der Verbindung ausgelöst worden.

5. **P_UABind** (PRES User Abort Indication)
   Die Darstellungsschicht (z.B. PRES_B) zeigt hiermit den Abbruch der Verbindung dem darüberliegenden ACSE–Protokollautomaten an. Der Auslöser des Abbruchs ist der andere Teilnehmer (User_A) gewesen.

6. **P_PABind** (PRES Provider Abort Indication)
   Damit kann die Darstellungsschicht einen Abbruch der Verbindung auslösen. Die Benachrichtigung der beiden ACSE–Protokollinstanzen muß durch diese Schicht gewährleistet werden. Auslöser des Abbruch ist der Diensterbringer, die Darstellungsschicht.

Damit ergibt sich der in Abbildung 6 gezeigte Zustandsübergangsgraph. „ABTSTATE“ bezeichnet eine Zustandsmenge, in der außer IDLE alle anderen möglichen Zustände des Automaten enthalten sind.

In Abbildung 7 ist das entsprechende Zeitdiagramm für einen Abbruch durch einen Dienstnutzer (z.B. User_A) bzw. in Abbildung 8 durch die Darstellungsschicht gezeigt.

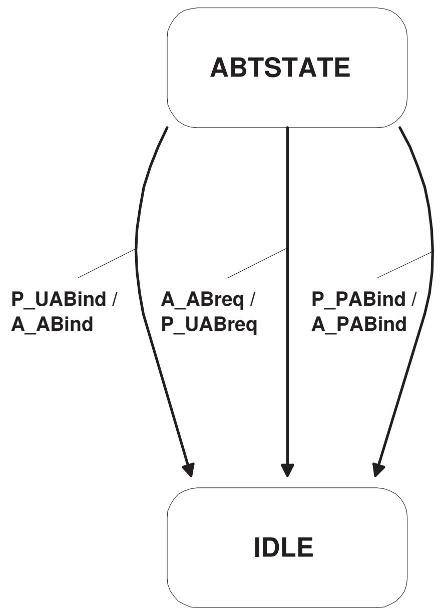

Abbildung 6: Zustandsübergangsgraph für den Abbruch einer Verbindung

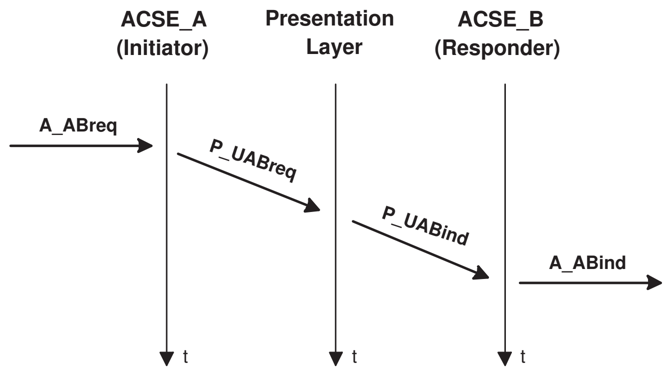

Abbildung 7: Abbruch einer Verbindung durch Dienstnutzer

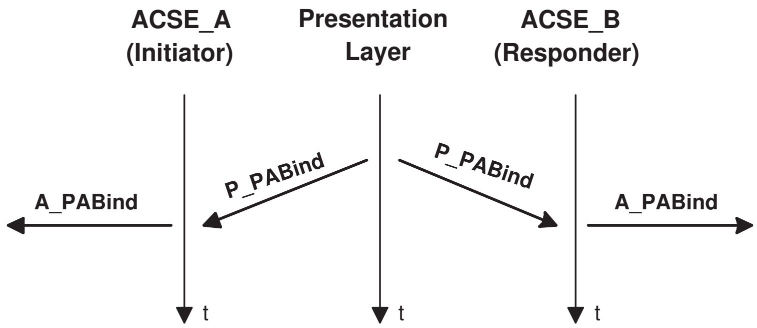

Abbildung 8: Abbruch einer Verbindung durch Darstellungsschicht

3 Spezifikationsstile

In diesem Kapitel werden einzelne Spezifikationsstile in der Spezifikationssprache Estelle anhand des ACSE–Dienstes betrachtet. Die einzelnen Stile sind:

Stil 1: Zustandsorientierter Stil

Stil 2: Nachrichtenorientierter Stil

Stil 3: Nachrichtenorientierter Stil ohne expliziten Kontrollzustand[9]

Stil 4: Nachrichtenorientierter Stil mit unifizierten[10] Nachrichten

Stil 5: Prozeduraler Stil[11]

Die zustandsorientierte und die nachrichtenorientierte Stilrichtungen können als klassische Stile in der Spezifikationssprache Estelle bezeichnet werden (siehe [11]). Klassisch deshalb, weil diese beiden Stilrichtungen die gebräuchlichsten Stile sind, ein Problem zu spezifizieren. Der Unterschied zwischen den beiden Stilrichtungen liegt darin, daß im zustandsorientierten Stil die Zustände, repräsentiert durch die „From"–Klausel in Estelle, im Vordergrund stehen, während im nachrichtenorientierten Stil die Nachrichten, repräsentiert durch die „When"–Klauseln, entsprechend vor den Zuständen aufgeführt werden.

Die nachrichtenorientierte Stilrichtung ohne expliziten Kontrollzustand ist aus dem zweiten Stil durch das Ablegen der Kontrollzustände in einer Variablen entstanden. Dadurch können die „From"–Schaltbedingungen einer Transition in den Transitionsblock verlagert werden. Dies ermöglicht einen direkten Zugriff auf den aktuellen Zustand des Automaten. Darüber hinaus dient diese Stilrichtung als Zwischenschritt für den nachrichtenorientierten Stil mit unifizierten Nachrichten und den prozeduralen Stil.

Das Hauptmerkmal des nachrichtenorientierten Stiles mit unifizierten Nachrichten ist, daß pro Interaktionspunkt die ursprünglichen Nachrichten in jeweils einen varianten Record für die zu sendenden bzw. für die zu empfangenen Nachrichten zusammengefaßt sind. Die Records werden dann als Parameter von neuen Nachrichten gesendet bzw. empfangen. Für diese Stilrichtung ist das Ablegen der Kontrollzustände in einer Variablen nicht erforderlich. Es läßt sich aber die Anzahl der Estelle–Transitionen auf zwei Transitionen reduzieren und dadurch die Auswahlzeit für eine Transition verkürzen.

Dagegen ist für den prozeduralen Stil das Ablegen der Kontrollzustände in einer Variablen notwendig. Diese Stilrichtung zeichnet sich dadurch aus, daß die zu empfangenen Nachrichten durch Prozeduren und die im Transitionsblock zu sendenden Nachrichten durch Prozeduraufrufe realisiert werden. Dadurch läßt sich die laufzeitineffiziente Kommunikation über Interaktionspunkte zwischen einzelnen Estelle–Modulen durch effizientere Prozeduraufrufe ersetzen. Prinzipiell ließen sich, statt der Nachrichten, die Zustände in Prozeduren umsetzen. Dies hätte aber den

[9] Die Zustände sind in einer Variablen abgelegt.

[10] Die verschiedenen Nachrichtentypen werden dabei zu zwei neuen Nachrichtentypen zusammengefaßt.

[11] Nachrichten sind in dieser Stilart als Prozeduren realisiert

Nachteil, daß für jede in einem Zustand empfangbare Nachricht eine Möglichkeit zur Übergabe der Nachrichtenparameter an die Prozedur, zum Beispiel durch jeweils ein zusätzliches Argument in den Prozedurköpfen, existieren müßte. Der prozedurale Stil geht auf einen Vorschlag von Jan Bredereke (siehe [1]) zurück.

Diese 5 verschiedenen Stile werden in den weiteren Abschnitten und Kapiteln unter den Gesichtspunkten

- der Lesbarkeit der einzelnen Stile,
- möglicher und notwendiger Erweiterungen von Estelle
- Unterscheidungsmerkmalen zwischen den verschiedenen Stilen und
- möglicher Unterschiede in der Laufzeiteffizienz zwischen den Stilen

untersucht.

Dazu sind 6 verschiedene Varianten des ACSE–Protokollautomaten als Estelle–Module angelegt worden. Dabei dient Variante 0 als Ausgangspunkt und Vergleichsbasis für die anderen Varianten. Die einzelnen Stile sind entsprechend den Varianten zugeordnet, d.h. Variante 1 entspricht dem Stil 1 usw. Die einzelnen Varianten sind als Instanzen der einzelnen Stilrichtungen zu verstehen und werden in Abschnitt 3.3 noch genauer besprochen.

Zuerst einige Bemerkungen zur Lesbarkeit und Laufzeiteffizienz.

3.1 Allgemeines zur Lesbarkeit und Laufzeiteffizienz

„Lesbarkeit“ ist ein subjektiver Begriff und läßt sich schlecht präzisieren. Die folgende Definition ist ein Versuch die Lesbarkeit einer Estelle–Spezifikation zu fassen und ist als Grundlage für alle weiteren Betrachtungen zur Lesbarkeit zu verstehen.

Definition (Gütemaß für Lesbarkeit) *Eine Spezifikation ist als „gut lesbar“ einzustufen, wenn die zu Grunde liegenden Sachverhalte, Strukturen und Abläufe im Spezifikationstext einfach zu erkennen bzw. zu verstehen sind.*

Unter Strukturen und Sachverhalte in einer Estelle–Spezifikation sind die Architektur bzw. Module der Spezifikation, d.h. die Protokollautomaten bzw. ihre Umsetzung in Estelle, zu verstehen. Damit verbunden sind die Abläufe innerhalb einer Transition, zwischen Transitionen bzw. zwischen den verschiedenen Modulen.

Daraus ergibt sich, daß eine Estelle–Spezifikation als „gut“ lesbar einzustufen ist, wenn aus dem Spezifikationstext, die Transitionen und Module als auch ihre jeweilige Aufgaben innerhalb eines Moduls bzw. des gesamten Systems relativ einfach heraus destilliert werden können. Sind zum Beispiel die endlichen, erweiterten Automaten, beschrieben durch Estelle–Module und Estelle–Transitionen, leicht aus der Spezifikation als Zustandsübergangsgraph zurück zu gewinnen, dann kann man die Spezifikation prinzipiell als „gut“ lesbar einstufen.

In der Definition des Gütemaßes für die Lesbarkeit ist implizit die Komplexität eines Spezifikationstextes und damit die Komplexität des Ausgangsproblems enthalten. Für eine weitere Präzisierung der Lesbarkeit müßte an dieser Stelle ein Bezugspunkt für die Komplexität auf Basis der einfachsten Estelle–Spezifikation definiert werden. Darauf aufbauend könnte die Definition der Komplexität einer allgemeinen Spezifikation erfolgen. Daraus ließe sich dann eine bessere Definition der Lesbarkeit, im Sinne einer „Reduzierung“ der Komplexität einer Spezifikation, herleiten.

Dies würde aber die vorliegende Arbeit sprengen und deshalb sind statt dessen folgende Kriterien und Regeln angegeben, die eine Verbesserung der Lesbarkeit zur Folge haben. Das Hauptaugenmerk liegt auf einer Verbesserung der Lesbarkeit innerhalb eines Moduls.

1. Allgemeingültige Regeln und Konventionen:
   (a) Die Bedeutungen der Nachrichten und Zuständen müssen aus den entsprechenden Bezeichnungen leicht zu erkennen sein. Für die Bezeichnungen von Modulen gilt entsprechendes.
   (b) Kommentare sollten das Wichtige und nicht sofort Offensichtliche erklären. Unnötige Kommentare, wie zum Beispiel „inc(x); {* x:=x+1 *}“, sind wegzulassen, da sie vom Wesentlichen nur ablenken.
   (c) Leerzeilen und spezielle Kommentarzeilen sollten eine Hervorhebung der Struktur bewirken.
2. Regeln für den Aufbau einer Estelle–Transition:
   (a) Der Aufbau einer Estelle–Transition sollte, soweit dies möglich ist, für alle Transitionen gleich sein.[12]
   (b) „Einfache“ Estelle–Transitionen mit gleichen Schaltbedingungen sollten zu einer „geschachtelten“ Transition zusammen gefaßt werden, d.h. die in mehreren Estelle–Transitionen vorhandenen, gleichen Schaltbedingungen tauchen im Idealfall nur noch einmal auf.
   (c) Bedingungen, zum Beispiel welche Nachricht von einer Estelle–Transition empfangen werden kann, müssen gut ersichtlich sein und im Vordergrund stehen.
   (d) Die Verschachtelungstiefe bestimmter Konstrukte (z.B. If–then–If–.. Ketten) sollte nicht zu Tief sein, da dies in drastischer Weise die Lesbarkeit bzw. ein einfaches Verständnis des Ablaufes verringert.
   (e) Eine sinnvolle Einrückung, die das Wesentliche unterstreicht und bereits optisch vom „Unwesentlichen“ trennt, ist zu wählen.

[12]Ein hoher Grad an Ordnung fördert das Verständnis und unterstreicht das Wesentliche

3. Regeln für ein Modul:

   (a) Alle zu einer Nachricht bzw. zu einem Zustand gehörenden Transitionen sollten nach Möglichkeit zusammenhängend dargestellt werden, um einen guten Überblick über die relevanten Sachverhalte zu erhalten.

   (b) Transitionen sollten nach Möglichkeit zu Gruppen zusammen gefaßt werden, wenn sie einen gemeinsamen Teilaspekt des Automaten realisieren.

   (c) Immer wiederkehrende, für den Ablauf des Automaten weniger wichtige Dinge, sollten in Prozeduren bzw. Funktionen ausgelagert werden. Dabei müssen aber geeignete Namen für die Prozeduren und Funktionen gewählt werden, so daß bereits aus dem Namen erkenntlich ist, zu welchem Zweck sie dienen.

   (d) Alle für ein Modul nur lokal benötigten Angaben, wie zum Beispiel lokale Variablen und Typen, sollten innerhalb des Moduls definiert werden um Namenskonflikte und Seiteneffekte zu verhindern.

4. Konventionen und Regeln für ein System:

   (a) Module sollten nach Möglichkeit jeweils in einer eigenen Datei realisiert werden[13]. Dies erhöht wesentlich die Wartbarkeit und Lesbarkeit, da bei verschachtelten Modulen, die gerade aktuelle Schachtelungstiefe nicht aus den Augen verloren geht.

   (b) Modulköpfe, Kanaldefinitionen etc. für die Sohnmodule sollten an einer Stelle im Vatermodul gesammelt werden. Änderungen zum Beispiel an der Modul–Attributierung lassen sich dadurch einfach bewerkstelligen.

Prinzipiell kann man sagen, daß diese Kriterien je nach Sichtweise auf ein System von unterschiedlicher Bedeutung sind und nur als Leitfaden für eine gut lesbare Estelle–Spezifikation dienen sollen.

Maßzahlen, die eine grobe Einschätzung der Lesbarkeit eines Estelle–Moduls ermöglichen, sind die Anzahlen von

- Kanten des Zustandsübergangsgraphen
- Nachrichtentypen
- Estelle–Transitionen
- Codezeilen
- Kommentar– und Leerzeilen

Für die Variante 1 bis Variante 5 gilt, daß sich die Anzahl der Transitionen sich der Anzahl von Nachrichtentypen annähert, während die Anzahl von Kanten im Zustandsgraphen konstant ist.

[13] Mittels „#include"–Befehl sind diese dann leicht einzubinden.

Weiterhin gilt, daß die Anzahl der verschiedenen Nachrichtentypen eine Untergrenze für die Anzahl von Estelle–Transitionen bildet, denn für jeden Nachrichtentyp, der eine oder mehrere Kanten im Zustandsgraphen darstellt, muß für eine korrekte Spezifikation mindestens eine Estelle–Transition existieren. Die Anzahlen für die Code–, Kommentar– und Leerzeilen dagegen variieren von Variante zu Variante nur unwesentlich. In Abschnitt 3.4 sind die entsprechenden Werte für die einzelnen Varianten aufgeführt.

Der Lesbarkeit gegenüber steht die Laufzeiteffizienz einer Spezifikation. Zum Beispiel sind Prozeduren und Funktionen ein gutes Hilfsmittel die Komplexität einer Transition bzw. eines Moduls zu verringern, führen aber im Allgemeinen zu einer Verschlechterung der Laufzeiteffizienz durch kopieren von Argumenten auf den Stack etc. Deshalb ist es je nach Schwerpunkt notwendig, geeignete Kompromisse zwischen Lesbarkeit und Laufzeiteffizienz zu schließen.

In Kapitel 4 sind die Meßwerte aus der Laufzeitmessung für den Auf– und Abbau von Verbindungen aufgeführt.

3.2 Gemeinsame Stilmittel, Konventionen

Die Namenskonventionen, die in allen Varianten gelten, sind in Kapitel 2.1 aufgeführt.

Ansonsten hat sich als sehr vorteilhaft[14] die Trennung der einzelnen bzw. von Gruppen von Transitionen durch unterschiedlich lange

```
{-----------------------------------------------}
```

Linien herausgestellt, da sie optisch die Struktur der Transitionen bzw. das Gruppieren der Transitionen unterstützen.

Weiterhin sind die Einrückungen so gewählt, daß zusammengehörige Dinge, zum Beispiel „Provided"–Klauseln, die gleiche Einrückungstiefe erhalten. Die Einrückungstiefe richtet sich wesentlich nach der Länge der Zeilen. Deshalb ist es nicht möglich gewesen, für alle Varianten eine gleiche Einrückungstiefe zu erreichen.

Um die Struktur des Automaten besser heraus zu arbeiten, sind die Transitionen entsprechend den drei Teilen des ACSE–Dienstes geordnet, soweit dies sinnvoll möglich ist[15].

Außerdem sind in allen Varianten die Schaltbedingungen vor den Schaltwirkungen aufgeführt und die Schaltbedingungen werden ab Variante 3 zum Teil durch ein „Case"–Konstrukt realisiert (siehe 3.3.4).

Schließlich hat sich die Maßnahme Teile der Transitionsblöcke in Prozeduren auszulagern ab Variante 1 als eine Verbesserung der Lesbarkeit herausgestellt. Dies wird in Abschnitt 3.3.2 noch genauer besprochen.

[14] Besonders beim Suchen einer Transition
[15] Für Variante 4 ist eine solche Anordnung nicht sinnvoll möglich.

3.3 Die einzelnen Varianten

Die im Folgenden besprochenen Varianten bzw. Stile lassen sich in zwei Klassen einteilen:
In die Klassen der

- zustandsorientierten bzw.
- nachrichtenorientierten Varianten

Die beiden Klassen unterscheiden sich durch die jeweilige Sichtweise auf den erweiterten, endlichen Automaten. Bei einer zustandsorientierten Sichtweise stehen die Zustände im Vordergrund. Sieht man dagegen den erweiterten, endlichen Automaten als Hilfsmittel zur Synchronisation des Empfangs bzw. des Sendens von Nachrichten an, dann ergibt sich die nachrichtenorientierte Sichtweise.

Estelle bietet durch das Anordnen der „From“– und „When“–Klauseln die Möglichkeit, die entsprechende Sichtweise zu realisieren. Im zustandsorientierten Stil tritt die „From“–Klausel vor den „When“–Klauseln und im nachrichtenorientierten Stil entsprechend umgekehrt auf. Darüber hinaus bietet Estelle die Möglichkeit beide Stilrichtungen innerhalb eines Moduls zu mischen. Dies kann bei großen Spezifikationen mehr zur Verwirrung als zur Klärung der Sachverhalte beitragen und beeinträchtigt damit die Lesbarkeit der Spezifikation. Deshalb sollte eine Mischung der Stile mit Vorsicht angewendet werden.

Die Entscheidung, welcher Stil für eine Estelle–Spezifikation besser geeignet ist, kann man danach treffen, in wievielen verschiedenen Zuständen eine Nachricht empfangbar ist und wieviele verschiedene Nachrichten pro Zustand empfangen werden können.

Es sei:

$$\mathbf{N} = \frac{\sum_{i=1}^{vZ} vN_i}{vZ}$$

$$\mathbf{Z} = \frac{\sum_{i=1}^{vN} vZ_i}{vN}$$

mit

vZ : Anzahl der verschiedenen Zustände
vN_i: Anzahl verschiedener, empfangbarer Nachrichten im Zustand i
vN : Anzahl der verschiedenen, empfangbaren Nachrichten
vZ_i: Anzahl verschiedener Zustände für eine empfangbare Nachricht i

Dann ergibt sich die in Tabelle 1 aufgeführte Richtschnur für die Wahl der Stilrichtung.

$\frac{N}{Z}$:	$\ll 1$	≈ 1	$\gg 1$
nachrichtenorientiert	$+$	$+/-$	$-$
zustandsorientiert	$-$	$+/-$	$+$

Tabelle 1: Richtschnur für die Wahl der Stilrichtung

Folgende Beispiele erläutern diesen Zusammenhang:

1. Fall: $(\frac{N}{Z} = \frac{n}{1} \gg 1)$

```
FROM 1                  WHEN 1
    WHEN 1                  FROM 1
      ⋮                 WHEN 2
    WHEN n                  FROM 1
                        WHEN 3
                          ⋮
                        WHEN n
                            FROM 1
```

2. Fall: $(\frac{N}{Z} = \frac{1}{n} \ll 1)$

```
FROM 1                  WHEN 1
    WHEN 1                  FROM 1
FROM 2                        ⋮
    WHEN 1                  FROM n
FROM 3
  ⋮
FROM n
    WHEN 1
```

Man erkennt, daß im Fall (1) ein zustandsorientierter Stil und im Fall (2) ein nachrichtenorientierter Stil besser geeignet ist.

Weiterhin zeigen diese Beispiele, wie aus „einfachen" Estelle–Transitionen „geschachtelte" Transitionen entstehen, und wie die Lesbarkeit durch geschachtelten Transitionen verbessert werden kann.

Für das ACSE–Modul sind die Werte für $\mathbf{\frac{N}{Z}}$ in Tabelle 2 gegeben. Es ist „ABTSTATE" die in Kapitel 2 erläuterte Zustandsmenge und wird bei der Berechnung als ein „Zustand" (mit) oder als entsprechende Anzahl von Zuständen (=7 ohne) interpretiert.

	ohne ABTSTATE	mit ABTSTATE
ohne Fehlerbehandlung	1.42	1.32
mit Fehlerbehandlung	1.38	1.26

Tabelle 2: Werte von $\frac{\mathbf{N}}{\mathbf{Z}}$ für den ACSE–Protokollautomaten.

Man stellt fest, daß ein zustandsorientierter Stil leicht besser zu sein scheint als ein nachrichtenorientierter Stil. Diese Aussage gilt nur für Variante 0 bis Variante 2.

Die anderen Varianten zeichnen sich, wie zum Beispiel Variante 3, dadurch aus, daß sie nur noch einen, nicht explizit aufgeführten Hauptzustand besitzen, d.h. es werden keine „From“–Klauseln mehr angegeben.

Für Variante 4 hat sich außerdem die Anzahl der relevanten Nachrichtentypen auf zwei reduziert. Damit können zwei verschiedene Nachrichten pro Zustand empfangen werden und nicht, wie in Variante 0 bis Variante 2, 11 verschiedene Nachrichten. Der Wert[16] für Variante 4 beträgt $\frac{N}{Z} = \frac{2}{8} = 0.25$.

Variante 5 besitzt gegenüber allen anderen Varianten auch keine „When“–Klauseln und kann deshalb mit der oben genannten Richtschnur nicht erfaßt werden.

In den folgenden Abschnitten wird auf die einzelnen Varianten bzw. Stile näher eingegangen. Das Augenmerk liegt auf den jeweiligen Stilmerkmalen und auf mögliche Estelle–Erweiterungen zur Verbesserung der Lesbarkeit.

3.3.1 Variante 0 (Ausgangspunkt)

Die Variante 0 dient für die folgenden Betrachtungen als Ausgangspunkt bzw. Vergleichsbasis. Sie ist aus einem zur Verfügung gestellten, fehlerhaften Quelltext entstanden.

Es mußten, um der Norm zu entsprechen bzw. unspezifizierten Empfang zu vermeiden, der fehlende Zustand „COLLISION_ASSOC_RESPONDER“ und die Estelle–Transitionen für das Erreichen und Verlassen des Zustandes hinzugefügt werden.

Außerdem sind unnötige Kopien von Estelle–Transitionen, die mehrfach im Quelltext vorkommen, aus dem Spezifikationstext entfernt worden.

Alle Angaben, die sich auf eine spezielle Umgebung auf Schicht 7 beziehen, sind in Variante 0 nicht mehr vorhanden. Für das korrekte Verhalten des ACSE–Protokollautomaten sind diese Angaben nicht notwendig.

Alle, zu Debugging–Zwecken im ursprünglichen Quelltext vorhandenen „writeln“–Ausgaben, sind herausgenommen worden, da sie vom Wesentlichen ablenken und für das korrekte Verhalten nicht notwendig sind.

[16] mit Fehlerbehandlung

Die Variante weist wie im Original die in Kapitel 2 erwähnte Unterteilung der Funktionalität in einen Teil für Aufbau, Abbau und Abbruch einer Verbindung und zusätzlich einen Teil für die Behandlung von Empfangsfehlern auf. Die einzelnen Transitionen sind entsprechend der vier Teile angeordnet.

Die im ursprünglichen Quelltext angewendete Möglichkeit, in Estelle mehrere Zustände zu einer Zustandsmenge („ABTSTATE") zusammen zu fassen, ist in den Teilen für den Abbruch und die Fehlerbehandlung beibehalten worden. Dadurch ist es möglich, viele einfache Estelle–Transitionen, die sich nur in der „From"–Schaltbedingung unterscheiden, zu einer geschachtelten Estelle–Transition zusammen zu fassen. Dies hat aber den Nachteil, daß den geschachtelten Transitionen Prioritäten zugeordnet werden müssen, da sonst in einem Zustand, durch Überlappung mit der Zustandsmenge, für eine Nachricht immer zwei schaltbare Estelle–Transitionen existieren würden. Deshalb sind alle Estelle–Transitionen für die Fehlerbehandlung mit einer niedrigeren Priorität versehen. Dies gewährleistet, daß beim Empfang einer Nachricht im korrekten Zustand die entsprechende Transition und nicht die Estelle–Transition für einen Empfangsfehler ausgeführt wird.

Weiterhin ist die Formatierung des ursprünglichen Quelltextes nach den oben erwähnten allgemeinen Stilregeln überarbeitet worden. Die Lesbarkeit hat sich dadurch verbessert.

Die angewendete Stilrichtung entspricht dem klassischen, zustandsorientierten Stil und wird im nächsten Abschnitt genauer besprochen.

3.3.2 Variante 1 (Zustandsorientiert)

Die Stilrichtung von Variante 1 entspricht dem „klassischen" zustandsorientierten Stil. Dieser Stil zeichnet sich dadurch aus, daß der Zustand, in dem sich der zu beschreibende erweiterte endliche Automat befindet, im Vordergrund steht, d.h. noch bevor die in diesem Zustand zu empfangenden Nachrichten beschrieben werden.

Die wesentliche Änderung von Variante 1 gegenüber Variante 0 ist das Herauslösen und Ersetzen von Teilen der Transitionsblöcke durch Prozeduren[17]. Dabei ist darauf zu achten, daß Informationen, die wesentlich den Ablauf einer Transition bestimmen, nicht mit in die Prozeduren aufgenommen werden. Sonst ist der Ablauf einer Transition nicht einfach nachzuvollziehen und verschlechtert damit die Lesbarkeit der Transition.

Wiederholt sich dagegen der Ablauf in einem Transitionsblock in mehreren anderen Transitionsblöcken, dann ist es unter Umständen besser, wichtige Informationen mit in Prozeduren aufzunehmen, um die Komplexität des Problems zu verringern.

Wichtig ist, daß aus den Namen der Prozeduren auf deren Aufgabe geschlossen werden kann. Die Prozeduren entsprechen hier einer Umsetzung einer Nachricht von einem Typ (empfangene) in eine Nachricht eines anderen Typs (zu sendende). Aus den ursprünglichen Transitionsblöcken werden dazu die Zuweisungen übernommen.

Die Prozeduren sind nach folgender Konvention benannt und aufgebaut:

[17] Die Prozeduren liegen in der Datei „Proc.e" vor.

```
procedure <empfangen>_to_<zu senden>( var <Quelle>;
                                      var <Ziel>;
                                      var <Sonstiges> );
begin
    <Block>
end;
```

mit

<empfangen>	bezeichnet die empfangende Nachricht
<zu senden>	bezeichnet die zu sendende Nachricht
<Quelle>	Parameter der empfangenen Nachricht
<Ziel>	Parameter der zu sendenden Nachricht
<Sonstiges>	sonstige Argumente
<Block>	Teile des ursprünglichen Transitionsblockes

Durch die Übergabe der Nachrichtenparameter an die Prozeduren mittels „Call–By–Reference“ Aufrufkonvention, kann der Nachteil einer Verschlechterung der Laufzeiteffizienz durch das Kopieren der Parameter auf den Stack beim Prozeduraufruf vermieden werden. Die „Call–By–Reference“ Aufrufkonvention schützt aber nicht davor, daß aus Versehen innerhalb des Prozedurblockes die Eingabeparameter geändert werden.

Die Struktur einer Transition sieht wie folgt aus:

```
{------------------------------------------------------------}
trans
priority <Priorität>
from     <Zustand>
         when <Nachricht 1>
              provided <Bedingung 1.1>
              to       <Folgezustand 1.1>
              begin
                       <Transitionsblock 1.1>
                       output <Folgenachricht 1.1>
              end;
               ⋮

         {------------------------------------------}
         when <Nachricht 2>
           ⋮

{------------------------------------------------------------}
trans
  ⋮
```

Man erkennt, daß die empfangbaren Nachrichten („When"–Klauseln) den Zuständen („From"–Klauseln) untergeordnet sind. Dadurch lassen sich alle empfangbaren Nachrichten nach den jeweiligen Zuständen gruppieren. Dies hat die Vorteile, daß Änderungen, Erweiterungen am Automaten bzw. am Verhalten des Dienstes einfacher bewerkstelligt werden können, da nur in den entsprechenden Abschnitten etwas geändert werden muß, und daß die Vollständigkeit[18] besser überprüft werden kann.

Eine Ausnahme beim Zuordnen der Nachrichten zu den entsprechenden Zuständen bildet im ACSE–Protokollautomaten der Teil für die Behandlung von möglichen Empfangsfehlern. Es ist nicht sinnvoll möglich, die benötigten Estelle–Transitionen den Zuständen und ihren regulär empfangbaren Nachrichten unter zu ordnen. Würde man dies dennoch tun, dann würde dies zu einer unnötigen Aufblähung der Spezifikation führen, da für jeden Zustand alle zu empfangenden Nachrichten aufgeführt werden müßten.[19]

Ein Kompromiß ist die bereits in Variante 0 und im ursprünglichen Quelltext angewendete Möglichkeit von Estelle, mehrere Zustände zu einer Menge zusammen zu fassen. Dieser Kompromiß hat aber die Nachteile, daß erstens Prioritäten eingeführt werden müssen, um Indeterminismen zu verhindern, und daß zweitens für jeden zu empfangenden Nachrichtentyp eine eigene Transition für die Behandlung von Empfangsfehlern erforderlich ist, die bis auf die „When"–Klauseln identisch sind.

Eine elegantere Lösung bietet sich an, wenn man die Möglichkeit hätte, mehrere Nachrichten zu einer Menge von Nachrichten, ähnlich der Menge von Zuständen, zusammen zu fassen. Damit bestände für die Fehlerbehandlung die Möglichkeit, die entsprechenden Estelle–Transitionen, je nach Semantik dieser Spracherweiterung, durch wenigere geschachtelte Transitionen auszudrücken. Die Einführung von Prioritäten ließe sich dadurch aber nicht verhindern.

Eine entsprechende Spracherweiterung sind die von R. L. Tenney (siehe [11]) vorgeschlagenen Interaktionsmengen.

Damit würde sich der gesamte Teil für die Fehlerbehandlung auf vier Estelle–Transitionen verkürzen:

```
{-------------------------------------------------------}
{*Definition der Interaktionsmengen im Deklarationsteil des Moduls*}
interactionset A_UnSpec = [ A_ASSreq, A_ASSrsp,
                            A_RELreq, A_RELrsp,
                            A_ABreq              ];

interactionset P_UnSpec = [ P_CONind, P_CONcnf,
                            P_RELind, P_RELcnf,
                            P_UABind, P_PABind   ];

   ⋮
```

[18] Vollständigkeit in dem Sinne, daß alle in einem Zustand benötigten Nachrichten spezifiziert sind.
[19] Dies wären pro Zustand 11 verschiedene Nachrichten !

```
{-----------------------------------------------------}
{*Transitionen für die Fehlerbehandlung*}
trans
priority PRIO_LOW
from     ABTSTATE
         when HIGH.A_UnSpec
              to     IDLE
              begin
                     InvalidBehaviour(a_a_abin, a_p_uabrq);
                     output HIGH.A_ABind;
                     output LOW.P_UABreq;
              end;
         {----------------------------------------}
         when LOW.P_UnSpec
            ⋮
{-----------------------------------------------------}
trans
priority PRIO_LOW
from     IDLE
         when HIGH.A_UnSpec
              to     IDLE
              begin
              end;
         {----------------------------------------}
         when LOW.P_UnSpec
            ⋮
```

Ein Vergleich mit dem Fehlerbehandlungsteil von Variante 1 zeigt sofort, daß sich die Anzahl von Transitionen wesentlich verringern würde, und daß sich dadurch die Lesbarkeit der Estelle–Spezifikation erhöhen ließe.

3.3.3 Variante 2 (Nachrichtenorientiert)

Die nachrichtenorientierte Stilrichtung von Variante 2 zeichnet sich dadurch aus, daß im Gegensatz zum zustandsorientierten Stil die Nachrichten im Vordergrund stehen. Dadurch lassen sich die Zustände und die Zustandsmenge „ABTSTATE“ den jeweiligen, empfangbaren Nachrichten zuordnen. Dies hat den Vorteil, daß ein eigener Abschnitt für die Fehlerbehandlung nicht gebildet werden muß, und daß alle durch eine Nachricht ausgelösten Aktionen zusammenhängend aufgeführt werden können.

Nachteilig an dieser Variante ist, daß die einzelnen Transitionen mit einer Priorität zu versehen sind, damit der, durch die Benutzung von „ABTSTATE“ eingeführte, Indeterminismus verhindert werden kann.

Allgemein läßt sich eine Transition in dieser Variante wie folgt beschreiben:

```
{------------------------------------------------------}
trans
when <Nachricht>
      priority <Priorität(Zustand 1)>
      from     <Zustand 1>
               provided <Bedingung 1.1>
               to       <Folgezustand 1.1>
               begin
                        <Transitionsblock 1.1>
                        output <Folgenachricht 1.1>
               end;
                 ⋮
      {----------------------------------------}
      priority <Priorität(Zustand 2)>
      from     <Zustand 2>
        ⋮
{------------------------------------------------------}
trans
  ⋮
```

Als besseres Verfahren mögliche Eingabefehler abzufangen, schlagen wir eine Änderung bzw. Erweiterung der Syntax der „From“–Klausel vor:

```
<from–clause>  ::= "from" <from–list>
<from–list>    ::= ["not"] <from–element>{","<from–element>}
                 | "otherwise"
<from–element> ::= <Zustand> | <Zustandsmenge>
```

mit der Semantik, daß die Transition ausgeführt wird, wenn der Automat sich nicht in dem durch „from“ spezifizierten Zustand bzw. nicht in einem durch die Zustandsmenge angegebenen Zustand befände, falls der optionale Parameter „not“ angegeben ist, und für „otherwise“, daß die Transition ausgeführt wird, wenn der Automat sich in einem Zustand befindet, der in den „From“–Klauseln bis zum letzten Auftreten der „Trans“–Anweisung nicht benutzt wird.

Die Vorteil dieser Syntax und Semantik wären, daß man besser die Zustände festlegen könnte, in denen die Transition schaltbar bzw. nicht schaltbar sein sollte, und man hätte durch „otherwise“ eine „Default“–Option für die „From“–Klauseln. Damit wäre es dann möglich, die Transitionen für die Fehlerbehandlung elegant ohne Zustandsmenge zu realisieren.

3.3.4 Variante 3 (Nachrichtenorientiert, ohne expliziten Hauptzustand)

Nachrichtenorientiert ohne expliziten Hauptzustand bedeutet, daß die Zustände, im Gegensatz zu den vorherigen Varianten, in einer einzigen Variablen kodiert sind und die Nachrichten, d.h. die „When“–Klauseln im Vordergrund stehen. Mit dieser Stilrichtung läßt sich die Anzahl der Transitionen auf die Anzahl empfangbarer Nachrichtentypen reduzieren.

Außerdem ist es möglich, die in Variante 2 angesprochene Estelle–Erweiterung der „From“–Klausel, indirekt zu realisieren. Dies kann durch eine „Case“–Abfrage über der Zustandsvariablen erfolgen.

Die Realisierung mit einer „Case“–Abfrage hat den Nachteil, daß jeder Zustand genau einmal in der Menge der „Case“–Alternativen auftauchen muß, da Estelle ISO–Pascal unterstützt und in dieser Sprache keine „Default“-Alternative für das „Case“–Konstrukt existiert (siehe [8]). Die Zustände, die zu einem Empfangsfehler führen würden, sind als letzte Alternativen im „Case“–Konstrukt aufgeführt.

Die Benutzung eines „Case“–Konstruktes macht es notwendig, die „Provided“–Klauseln durch „If–Then–Else“–Abfrageketten auszudrücken. Sind die Transitionsblöcke komplexer und besitzen weitere „If“–Abfrageketten, dann wird die Lesbarkeit eingeschränkt, da sich möglicherweise die Schaltbedingungen für einen Transitionsblock mit anderen internen Bedingungen vermischen können, d.h. die Schaltbedingungen sind schlechter zu erkennen.

Eine Maßnahme, die in dieser Variante eine gesteigerte Bedeutung in Bezug auf die Lesbarkeit besitzt, sind die gestrichelten Linien. Sie ermöglichen es, zusammengehörige Abschnitte schneller zu erkennen und mildern dadurch den vorher genannten Nachteil des „Case“–Konstruktes etwas ab.

Insgesamt hat sich die Lesbarkeit der Spezifikation gegenüber der Variante 2 verschlechtert, da nicht mehr einfach erkennbar ist, welcher Transitionsblock unter welchen Bedingungen bearbeitet wird.

Die offensichtlichen Vorteile dieser Stilrichtung sind, daß man bei Bedarf auf den Zustand auch innerhalb des Transitionsblockes zugreifen und daß man das Verhalten mit wenigeren Transitionen beschreiben kann. Dadurch ist es möglich, die Zeit für die Auswahl einer Estelle–Transition zu verkürzen.

Eine Transitionen besitzt in dieser Stilrichtung folgende Struktur:

```
{-------------------------------------------------------}
trans
when   <Nachricht>
begin  case (<Zustandsvariable>) of

       <Zustand 1>:
            begin
               if <Provided–Bedingung 1.1>
               then begin
                       <Zustandsvariable>:=<Folgezustand 1.1>;
                       <Transitionsblock 1.1>
                       output <Folgenachricht 1.1>
               end

               {---------------------------}
               else if <Provided–Bedingung 1.2>
               then ...
                 ⋮

               {---------------------------}
               else begin {*provided otherwise*}
                       <Zustandsvariable>:=<Folgezustand 1.n>;
                       <Transitionsblock 1.n>
                       output <Folgenachricht 1.n>
               end;
            end; {*Zustand 1*}

       {---------------------------------------}
       <Zustand 2>:
            ⋮
       end; {*case*}
end; {*when*}

{-------------------------------------------------------}
trans
  ⋮
```

3.3.5 Variante 4 (Nachrichtenorientiert mit unifizierten Nachrichten)

Die Variante 4 baut auf Variante 3 auf, d.h. sie entspricht einer nachrichtenorientierten Stilrichtung ohne expliziten Hauptzustand. Prinzipiell läßt sich diese Variante als zustandsorientierte Version realisieren, allerdings ist nach den Vorbemerkungen (siehe Abschnitt 3.3) ein nachrichtenorientierter Stil besser geeignet. Dieser entspricht auch genauer der Zielsetzung bzw. Aufgabe, die ein Estelle–Modul in dieser Stilart meistens zu realisieren hat.

Sinnvoll ist das Unifizieren von Nachrichten, wenn innerhalb des beschriebenen Dienstes sehr wenige der ursprünglichen Nachrichten in irgendeiner Form bearbeitet werden, während der größere Anteil der Nachrichten unbehandelt weitergereicht wird.

Der Unterschied zur Variante 3 liegt darin, daß die einzelnen Nachrichten unifiziert sind. Dies bedeutet eine Zusammenfassung der ursprünglichen Nachrichten in jeweils einen Varianten–Record für einen Interaktionspunkt. Dies hat für das ACSE–Beispiel die Konsequenz, daß zwei Variante–Records, jeweils einen für die Kommunikation mit dem Dienstnutzer und einen für die Nachrichten an die darunter liegende Darstellungsschicht, angelegt werden müssen.

Der Vorteil des Unifizierens ist, daß die Anzahl von Nachrichtentypen auf die Anzahl von Interaktionspunkten reduziert werden kann. Dadurch wird erreicht, daß die Auswahlphase für eine Transition kürzer wird. Dagegen hat dieser Stil den Nachteil, daß die einzelnen Transitionen durch das Unifizieren komplexer werden und die Lesbarkeit der Spezifikation deutlich abnimmt.

Weiterhin sollte man beim Unifizieren von Nachrichten die Frage nach der Größe der Nachrichten, d.h. die Größe der Nachrichtentypen, nicht unbeachtet lassen. Die Größe spielt dann eine Rolle, wenn der verwendete Estelle–Compiler für die Größe eines varianten Records die größte Alternative im Record zugrunde legt, und wenn beim Senden einer Nachricht immer diese maximale Größe des varianten Records übertragen wird. Für das Zusammenfassen der ursprünglichen Nachrichten bedeutet dies, daß die Größe der neuen Nachrichten jeweils durch die maximale Größe über den „alten Nachrichten“ bestimmt wird. Dies hat bei unterschiedlichen Größen der „alten Nachrichten“ zur Folge, daß beim Verschicken einer ursprünglich kleinen Nachricht unnötiger Ballast mit übertragen werden muß. Außerdem wird dadurch der Vorteil einer kürzeren Auswahlphase zunichte gemacht (siehe Kapitel 4). Als Beispiel sei eine Nachricht mit sehr vielen Parametern bzw. Daten und die dazugehörige Quittung mit sehr wenigen bzw. gar keinen Parametern genannt.

Ein weiterer Nachteil, der durch das Unifizieren in Kauf genommen werden muß, ist, daß die Kanaldefinition, wie zum Beispiel zwischen dem Nutzer und dem ACSE–Protokollautomaten bzw. zwischen dem ACSE–Protokollautomaten und der Darstellungsschicht, geändert werden müssen.

Damit ergibt sich für Variante 4 folgende neue Kanaldefinition:

```
{-------------------------------------------------------}
{*Typ–Definitionen*}
<NA_1> = (<N_1.1>,...,<N_1.n>);
<NT_1> = record
                case ( <N_IS>  : <NA_1>) of
                        <N_1.1> : (<PDU_1.1> : <PDU_Typ_1.1>);
                            ⋮
                        <N_1.n> : (<PDU_1.n> : <PDU_Typ_1.n>);
            end; {*record*}
{-------------------------------------------------------}
<NA_2> = ( ...
<NT_2> = record ...

{-------------------------------------------------------}
{*Kanal–Definitionen*}
channel <Kanal_Typ_1> (<Rolle_1>,<Rolle_2>);
by <Rolle_1>
             <Nachricht_1> (<MSG> : <NT_1>);
by <Rolle_2>
             <Nachricht_2> (<MSG> : <NT_2>);

{-------------------------------------------------------}
channel <Kanal_Typ_2> (<Rolle_1>,<Rolle_2>);
     ⋮
```

mit

<N_*>	:	„Alte“ Nachrichten
<NA_*>	:	Aufzählungstyp, über die „alten“ Nachrichten
<NT_*>	:	„Neue“ Nachrichtentypen (Varianter–Record)
<PDU_Typ_*>	:	„Alte“ Nachrichtentypen
<PDU_*>	:	Parameter der „alten“ Nachrichten
<MSG>	:	Parameter der „neuen“ Nachrichten
<N_IS>	:	Auswahl der „alten“ Nachricht

Die Struktur einer Transition für unifizierte Nachrichten sieht wie folgt aus:

```
{---------------------------------------------------------}
trans
when  <Nachricht>
begin
      case (<MSG.N_IS>) of
            <N_1.1>:
            begin case (<Zustandsvariable>) of
                  <Zustand 1>:
                      ⋮       „entspricht Variante 3“
            end; {*Nachricht N_1.1*}
            {-----------------------------------------}
            <N_1.2>:
                ⋮
      end; {*case Nachricht*}
end; {*when*}
{---------------------------------------------------------}
trans
  ⋮
```

Die Lesbarkeit einer Transition hat gegenüber den vorherigen Varianten deutlich abgenommen, da es verschachtelte „Case“–Konstrukte, einen Äußeren für die „alten“ Nachrichten und einen Inneren für die Zustände, gibt.

Eine Verbesserung der Lesbarkeit läßt sich erreichen, wenn das innere „Case“–Konstrukt jeweils in Prozeduren ausgelagert wird. Allerdings ergibt sich dann ein Problem mit den „Output“–Anweisungen, die nach Estelle (siehe [8]) ausschließlich im Transitionsblock angewendet werden dürfen und folglich nicht innerhalb von Prozeduren und Funktionen.

Lösungen für dieses Problem wären eine entsprechenden Änderung von Estelle (siehe [1]) oder zum Beispiel eine Übergabe der benötigten Nachrichtenparameter beim Prozeduraufruf mittels „Call–By–Referenze“–Konvention, so daß nach Beendigung der Prozedur die Nachrichtenparameter durch eine „Output“–Anweisung in den Alternativen des äußeren „Case“–Konstruktes verschickt werden könnten. Diese Lösung hat aber den Nachteil, daß für jeden Nachrichtenparameter ein eigenes Argument im Prozedurkopf angelegt werden muß.

Als Beispiel für eine mögliche Anwendung dieses Stiles sei die Spezifikation eines Terminals genannt (siehe [1]). In einer solchen Spezifikation müßte ohne das Unifizieren von Nachrichten jedes Zeichen von oder zum Terminal durch eine eigene Nachricht explizit definiert werden, obwohl nur sehr wenige Zeichen für die eigentliche Steuerung des Terminals benötigt werden, während der größere Teil weitergereicht wird.

3.3.6 Variante 5 (Prozeduraler Stil)

Der prozedurale Stil (siehe [1]) entsteht, wenn man die Transitionen in einem Estelle–Modul durch Prozeduren ersetzt.

Durch diese Maßnahme ist es möglich, Estelle–Module mit anderen Modulen, zum Beispiel das ACSE–Modul mit dem Modul für die Darstellungsschicht, zu verschmelzen. D.h. statt eine Nachricht über die entsprechenden Interaktionspunkte und die zugeordneten Warteschlangen durch eine „Output“–Anweisung zu verschicken, kann die zu einer Nachricht gehörende Prozedur direkt aufgerufen werden. Dadurch erreicht man, daß die laufzeitineffiziente Kommunikation zwischen Estelle–Modulen durch laufzeiteffizientere Prozeduraufrufe ersetzt wird.

Man beachte aber, daß das Austauschen die in Estelle benutzte asynchrone Kommunikation zwischen Modulen in eine synchrone Kommunikation wandelt. Ist in einer Estelle–Spezifikation die asynchrone Kommunikation für das korrekte Verhalten der Spezifikation unbedingt erforderlich, dann ist eine direkte Umsetzung von Estelle–Transitionen in Prozeduren nicht möglich. Es muß in diesem Fall der Asynchronität der Kommunikation, zum Beispiel durch das explizite Spezifizieren und Benutzen einer Warteschlange, Rechnung getragen werden.

Ein Problem, das in ähnlicher Form in der Variante 4 bereits auftaucht, sind „Output“–Anweisungen innerhalb von Prozeduren. Dieses Problem stellt sich dann, wenn zwischen zwei, auf verschiedenen Rechnern auszuführenden Modulen Nachrichten ausgetauscht werden sollen, d.h. wenn zwei Module nicht verschmolzen werden können bzw. sollen.

Prinzipiell könnte man die Kommunikation mit anderen Modulen dadurch realisieren, in dem man die verschmolzenen Module in ein weiteres Modul einbettet und die Nachrichtenparameter dort mit einer „Output“–Anweisung verschickt. Dazu müßten die Prozeduren eine Möglichkeit besitzen, zum Beispiel durch zusätzliche Argumente in den Prozedurköpfen, um die Nachrichtenparameter an das umgebende Modul übergeben zu können. Diese Lösungen sind inakzeptabel, weil die Lesbarkeit der Spezifikation wesentlich darunter leiden würde. Man stelle sich zum Beispiel eine Prozedur vor, die sehr viele Nachrichten an andere Module verschicken müßte, d.h. sie hätte sehr viele zusätzliche Argumente im Prozedurkopf.

Eine bessere Lösung ist die, von J. Bredereke (siehe [1]) vorgeschlagene, Änderung der Estelle–Syntax bzw. Semantik für „Output“–Anweisungen. Dieser Vorschlag sieht vor, daß innerhalb von Prozeduren „Output“–Anweisungen erlaubt sind.

Ein weiteres Problem beim Verschmelzen zweier Module sind Namenskonflikte, zum Beispiel für die in einem Modul lokal definierten Variablen, Konstanten oder Prozeduren und Funktionen. Deshalb ist es notwendig, diesen einen eindeutigen neuen Namen zu vergeben. Die in dieser Variante angewendete Methode ist, die neuen Namen jeweils mit dem Namen des Moduls als Präfix zu versehen. Sinnvoll ist auch, die Variablen eines Moduls in einem Record zusammen zu fassen und innerhalb der Prozeduren mittels der „With“–Anweisung auf die Komponenten dieses Records zu zugreifen.

Für die Prozeduren, Variablen und Konstanten[20] gilt folgende Vereinbarung:

ACSE_<Nachricht>**(...);**	Für die Prozeduren
ACSE_<Zustand>	Für die Zustände
ACSE_<GlobalVar>	Record für die globalen Variablen

Falls aus den Namen der Nachrichten nicht eindeutig hervorgeht, zu welchem ursprünglichen Interaktionspunkt sie gehören, sollten die entsprechenden Namen für die Prozeduren um einen Teil zur Kennzeichnung des entsprechenden Interaktionspunktes ergänzt werden.

Die Umsetzung der Estelle–Transitionen in Prozeduren kann nun nach folgendem Schema durchgeführt werden:

- Umwandlung der Estelle–Spezifikation in einen nachrichtenorientierten Stil, in dem bis auf die „When"–Klausel alle anderen Schaltbedingungen innerhalb des Transitionsblockes realisiert werden (siehe Variante 3). Ziel ist, daß nach der Umwandlung für jeden Nachrichtentyp genau eine Estelle–Transition, mit der „When"–Klausel als einzige Schaltbedingung, existiert.
- Für jede dieser transformierten Estelle–Transitionen ist eine Prozedur anzulegen. Der Prozedurkopf enthält ein Argument für die Parameter des Nachrichtentyps und der Prozedurrumpf ergibt sich aus dem Transitionsblock der transformierten Estelle–Transition.

Probleme bereiten, beim Umwandeln der Estelle–Transitionen in die erforderliche transformierte Form, spontane Estelle–Transitionen, Transitionen mit „Delay"–Klauseln, Prioritäten, Realisierung von gewünschtem Indeterminismus etc. Mögliche Lösungen sind zum Beispiel für die Prioritäten eine Realisierung durch „If"–Abfrageketten und für gewünschten Indeterminismus die „Forone"–Anweisung von Estelle. Für Lösungen zu den anderen Problemen sei auf [1] verwiesen.

Das ACSE–Beispiel läßt sich sehr einfach, wie Variante 3 zeigt, in die gewünschte Ausgangsform transformieren.

Auf der nächsten Seite ist der prinzipielle Aufbau der Prozeduren und der Aufbau der Transitionen, die benötigt werden, wenn verschmolzene Module mit anderen Modulen kommunizieren sollen, dargestellt. Es erweist sich als Vorteil, diese Transitionen in eine eigene Datei zu verlegen, weil dadurch das Verschmelzen mit weiteren Modulen vereinfacht wird und innerhalb der zu verschmelzenden Module wenigere Änderungen durchzuführen sind. Außerdem wird dadurch eine bessere Übersicht über verschmolzene und nicht verschmolzene Module gewährleistet.

---

[20] In dieser Variante sind die Konstanten des Aufzählungstyps für die Zustandsvariable, d.h. die Zustände, gemeint.

Aufbau der Prozeduren:

```
{----------------------------------------------------------}
procedure ACSE_<Nachricht>( var <Parameter> : <Parameter–Typ>);
begin
      with ACSE_<GlobalVar> do
      begin
            case (<Zustandsvariable>) of
            <Zustand 1>:
                 begin
                     if <Provided–Bedingung 1.1>
                     then begin
                                .
                                .
                                .
                            output <Folgenachricht 1.1>;
                            <Prozedur–Aufruf>(...);
                     end
                       .
                       .
                       .
                 end; {*Zustand 1*}
            {------------------------------------------}
            <Zustand 2>:
                   .
                   .
                   .
            end; {*case Zustand*}
      end; {*with*}
end; {*procedure*}

{----------------------------------------------------------}
procedure ...
     .
     .
     .
```

Aufbau der Transitionen:

```
{----------------------------------------------------------}
trans
when  <Nachricht>
        begin
                ACSE_<Nachricht>(...);
        end;

{----------------------------------------------------------}
trans
   .
   .
   .
```

Insgesamt hat sich die Lesbarkeit dieser Variante gegenüber den ersten drei Varianten verschlechtert, da die Struktur der Spezifikation, die Module bzw. Transitionen nicht mehr einfach erkennen lassen. Der Vorteil des prozeduralen Stils liegt dagegen in der höheren Laufzeiteffizienz (siehe auch Kapitel 4), wenn mehrere Module verschmolzen werden. Das Verschmelzen hat aber den Nachteil, daß eine parallele Ausführung der einzelnen Module nicht möglich ist. Für die Spezifikation des OSI–Schichtenmodells ist dies kein Nachteil, weil das Schichtenmodell eine sequentielle Abarbeitung mit synchroner Kommunikation nahelegt. Man könnte allerdings die einzelnen Schichten als Pipeline realisieren. Dann sind parallel laufende Module notwendig und folglich ist das Verschmelzen von Schichten hinderlich.

Für weitergehende Betrachtungen zu dieser Stilrichtung sei auf [1] verwiesen.

3.4 Zusammenfassung

Zusammenfassend kann man sagen, daß die Variante 2, d.h. der nachrichtenorientierte Stil, bzw. die Variante 1, d.h. der zustandsorientierte Stil, in Bezug auf die Lesbarkeit für die Spezifikation des ACSE–Protokollautomaten am Besten geeignet sind, da diese Varianten den meisten der in Abschnitt 3.1 aufgeführten Regeln bzw. Konventionen entsprechen. Für welche der beiden Stilrichtungen man sich entscheidet, ist abhängig von der jeweiligen Sichtweise auf den Automaten. Ein Hilfsmittel, die Entscheidung zwischen den beiden Stilen zu vereinfachen, ist die in Abschnitt 3.3 angegebene Richtschnur.

Gegenüber den Varianten 0 und 1 hat die Variante 2 den Vorteil, daß alle zu einer Nachricht gehörenden Aktionen der entsprechenden Nachricht untergeordnet sind und gegenüber den Varianten 3 bis 5, daß die Komplexität der einzelnen Transitionen geringer ist. Allerdings hat sie gegenüber der Variante 3 den Nachteil, daß sie Prioritäten benötigt, damit der durch die Zustandsmenge eingeführte Indeterminismus verhindert werden kann. Eine Abhilfe würde an dieser Stelle die Erweiterung der „From"–Klausel, wie in Variante 2 angesprochen, ermöglichen. Mit den „Interaction–Sets" wäre die Variante 1 in Bezug auf die Lesbarkeit eine sehr gute Wahl.

Muß man dagegen auf die Zustände innerhalb der Transitionsblöcke zugreifen, dann führt kein Weg daran vorbei, die Hauptzustände in einer Variablen unter zu bringen, wie zum Beispiel in Variante 3 geschehen.

Liegen die Nachrichtenpakete in der gleichen Größenordnung und werden in dem Modul nur sehr wenige dieser Pakete bearbeitet, dann ist das unifizieren der Nachrichten sinnvoll. Man kann in dieser Stilart sich auf die wenigen notwendigen Anweisungsfolgen beschränken und die für das Weiterreichen von Nachrichten benötigten Anweisungen, kompakt und elegant mittels des „Case"–Konstruktes abhandeln[21] Für Variante 4 des ACSE–Protokollautomaten ist dies nicht gegeben, da die Nachrichten zum größten Teil bearbeitet werden und die Größen der ursprünglichen Nachrichtenpakete sehr unterschiedlich sind.

Variante 5 ist in Bezug auf die Laufzeiteffizienz die am Besten geeignete Stilrichtung, da die aufwendige Kommunikation zwischen verschmolzenen Modulen durch laufzeiteffizientere Proze-

[21] Gäbe es eine „Default"–Alternative für das „Case"–Konstrukt, dann wäre die Abhandlung noch besser zu bewerkstelligen !

duraufrufe ersetzt werden kann (siehe Kapitel 4). Allerdings erkauft man sich in der prozeduralen Stilrichtung diese Effizienz auf Kosten der Lesbarkeit. Außerdem ändert man die Verbindung zwischen zwei Modulen beim Verschmelzen von einer asynchronen zu einer synchronen Verbindung. Weiterhin verhindert das Verschmelzen von Modulen eine parallele Ausführung der Module bzw. die Möglichkeit eine Pipeline zu etablieren.

Die in Abschnitt 3.1 angegebenen Maßzahlen für eine grobe Einschätzung der Lesbarkeit sind in Tabelle 3 und das Verhältnis der einzelnen Varianten zu Variante 0 in Tabelle 4 aufgeführt. Variante 5m ist entstanden durch eine Verschmelzung des ACSE–Protokollautomaten (Variante 5) mit der Darstellungsschicht (siehe Kapitel 4 und 3.3.6).

Variante:	0	1	2	3	4	5	5m
Kanten des Zustandsgraphen	22	22	22	22	22	22	22
Nachrichtentypen empfangbar	11	11	11	11	2	–	–
Nachrichtentypen gesamt	22	22	22	22	4	–	–
Estelle-Transitionen	40	40	40	12	3	12	–
Codezeilen ohne Prozeduren	–	351	388	367	394	466	385
Codezeilen gesamt	595	538	575	554	581	653	572
Kommentar–/Leerzeilen ohne Proz.	–	156	156	118	100	165	119
Kommentar–/Leerzeilen gesamt	162	235	235	197	179	244	198
Zeilen gesamt	757	773	810	751	760	897	770

Tabelle 3: Maßzahlen für eine grobe Einschätzung der Lesbarkeit

Variante * / Variante 0	1	2	3	4	5	5m
Nachrichtentypen gesamt	1	1	1	0,182	–	–
Estelle-Transitionen	1	1	0,3	0,075	0,3	–
Codezeilen gesamt	0,904	0,966	0,931	0,976	1,097	0,961
Kommentar–/Leerzeilen gesamt	1,450	1,450	1,216	1,104	1,506	1,222
Zeilen gesamt	1,021	1,070	0,992	1,004	1,184	1,017

Tabelle 4: Verhältnis der einzelnen Varianten zu Variante 0

Man erkennt, daß die einzelnen Varianten gegenüber der Variante 0 in der Anzahl von Codezeilen wenig variieren, da wenig Code im ACSE–Automaten wieder verwendbar ist. Dagegen zeichnen sich die Varianten 3 bis 5m durch wenige Estelle–Transitionen bzw. wenigeren Nachrichtentypen aus, sind aber in Bezug auf ihre Lesbarkeit (besonders Variante 4) schlechter als die Varianten 1 bis 2.

Als Folgerung aus den Maßzahlen und den vorher gemachten Aussagen ergibt sich für die Lesbarkeit, daß der zustandsorientierte bzw. nachrichtenorientierte Stil für die Spezifikation des

ACSE–Automaten am Besten geeignet sind. Dies ist nicht verwunderlich, da diese Stilrichtungen die speziellen Merkmale, z.B. „From“– und „When“–Klauseln, der Spezifikationssprache Estelle besonders gut ausnutzen.

4 Laufzeitmessung

Dieses Kapitel beschäftigt sich mit der Laufzeiteffizienz der einzelnen Stilrichtungen. Es stellt sich die Frage, in wieweit die Stilrichtungen einen Einfluß auf die Effizienz bzw. Ausführungszeit der einzelnen Varianten besitzen.

4.1 Meßaufbau

Der Meßaufbau für die verschiedenen Varianten ist in Abbildung 9 dargestellt. Das Estelle–

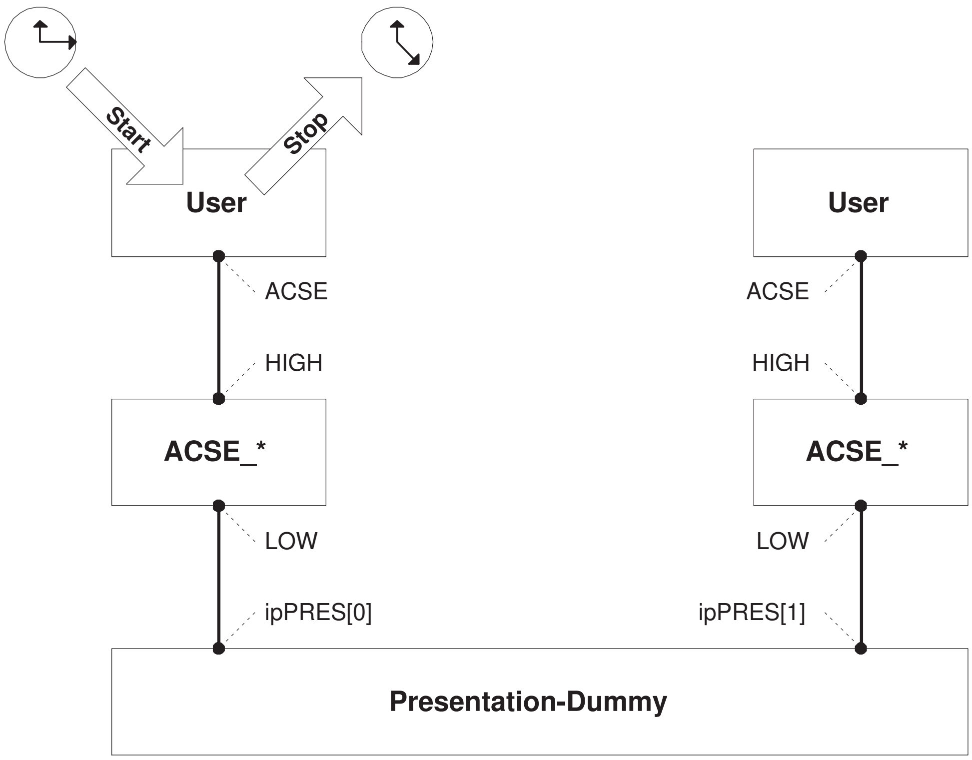

Abbildung 9: Meßaufbau zur Messung der einzelnen Stile

Modul „User“ dient als Sender und Empfänger der Verbindung und „Presentation–Dummy“ als Übertragungsschicht. Prinzipiell läßt sich mit diesem Aufbau nur das gesamte System messen und nicht die genauen Zeiten für die einzelnen Varianten.

Deshalb ist es notwendig den Anteil der Umgebung an der Messung herauszufinden. In Abbildung 10 ist der Meßaufbau für die Umgebung abgebildet. Allerdings ist diese Anordnung nicht unproblematisch, da die beiden Estelle–Modulen User und Presentation–Dummy nicht über ihre Interaktionspunkte miteinander verbunden werden können. Der Grund dafür sind die unterschiedlichen Nachrichten, die von den beiden Modulen empfangen bzw. gesendet werden können. Eine Lösung dieses Problems ist die Anpassung einmal der Schnittstelle des Nutzers

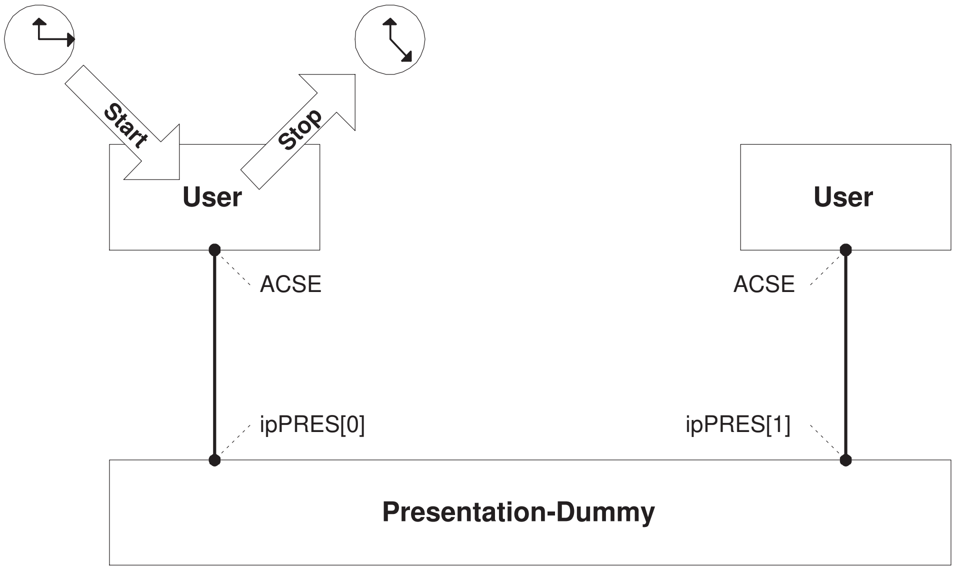

Abbildung 10: Meßaufbau für die Umgebung

an das Übertragungsmodul bzw. die Schnittstelle des Übertragungsmoduls an die des Nutzers. Damit ergeben sich zwei Messungen, aus denen, mit einem gewissen Fehler behaftet, die Zeit der Umgebung ermittelt werden kann (siehe Abschnitt 4.2).

Damit die Ungenauigkeiten nicht zu groß werden, ist die interne Struktur der beiden Module, d.h. die Abfolge der Transitionen und ihr Aufbau, nicht geändert worden. Daraus folgt, daß die Ungenauigkeiten auf die verschiedenen Größen der Nachrichtentypen zurückzuführen sind. Weiterhin besitzen die beiden Module nur absolut notwendige Transitionen für einen erfolgreichen Verbindungsauf–, Verbindungsabbau bzw. für das Messen notwendige Transitionen. Außerdem werden die zu übertragenden Datenpakete vor einer Messung mit dem notwendigsten initialisiert und auf der Übertragungsschicht nicht von einem Nachrichtentyp in einen anderen transformiert, sondern lediglich durch weitere, bereits zum Zeitpunkt der Messung initialisierte Pakete, ersetzt. Dadurch kann die Verweildauer in den User–Modulen und dem Übertragungsmodul auf ein Minimum gedrückt werden, wodurch die angesprochene Ungenauigkeit weiter verringert werden kann.

Für Variante 4 ist eine Änderung der Schnittstelle, wie in Abschnitt 3.3.5 beschrieben, zwingend notwendig und damit eine getrennte Ermittlung der Zeit für die Umgebung. Es wird dazu der gleiche Meßaufbau benutzt.

Ein anderes Verfahren wird für die Variante 5 benutzt. Die Variante existiert in zwei Versionen, die eine für den oben beschriebenen Meßaufbau mit den dafür notwendigen Transitionen, die andere Version[22] ist dagegen mit den Schichten 5 und 6 verschmolzen. Dies trägt dem Ziel des prozeduralen Stils besser Rechnung. Für das Verschmelzen sind, wie in Abschnitt 3.3.6

[22] Diese Version ist als Datei „acse_5m.e“ vorhanden.

aufgeführt, die entsprechenden Anweisungen geändert und die Transitionen für die Schnittstelle mit einem Nutzer in die Datei „protokoll.e“ ausgelagert worden.

Als Basis für einen Vergleich wird die Variante 0, eingebettet in den entsprechenden Protokollstack, herangezogen. In Abbildung 11 und 12 sind die entsprechenden Meßaufbauten dargestellt.

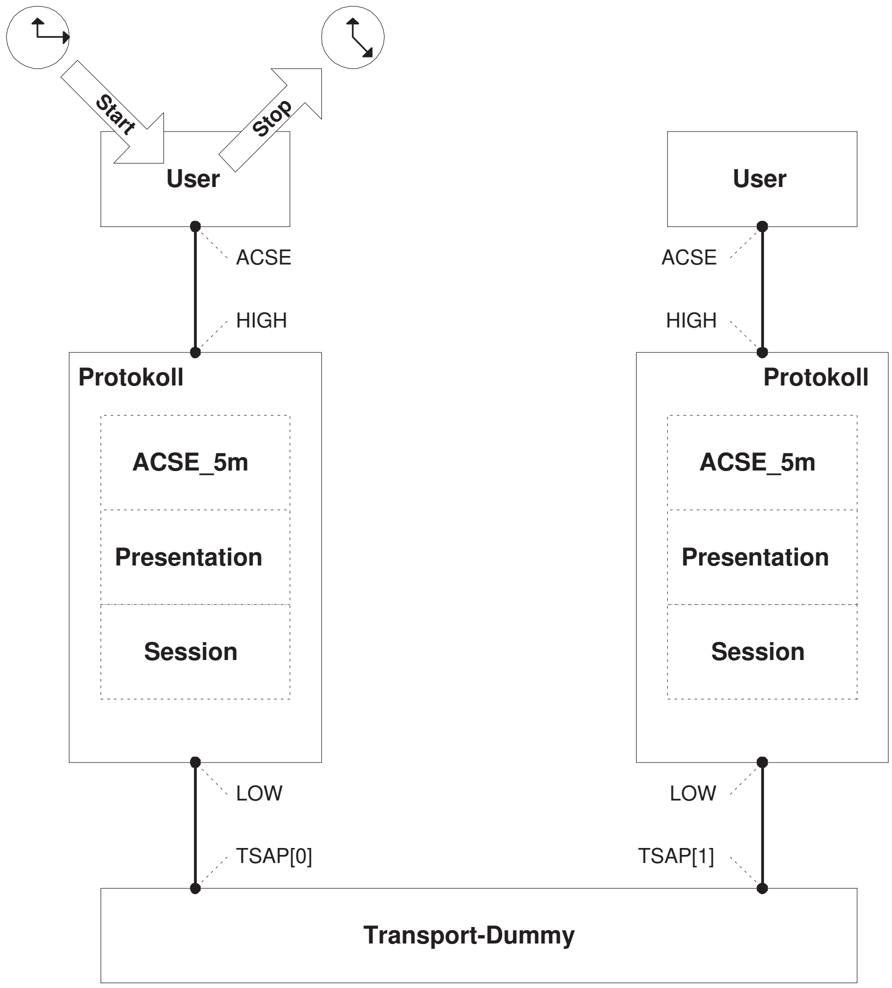

Abbildung 11: Meßaufbau für verschmolzenen Protokollstack

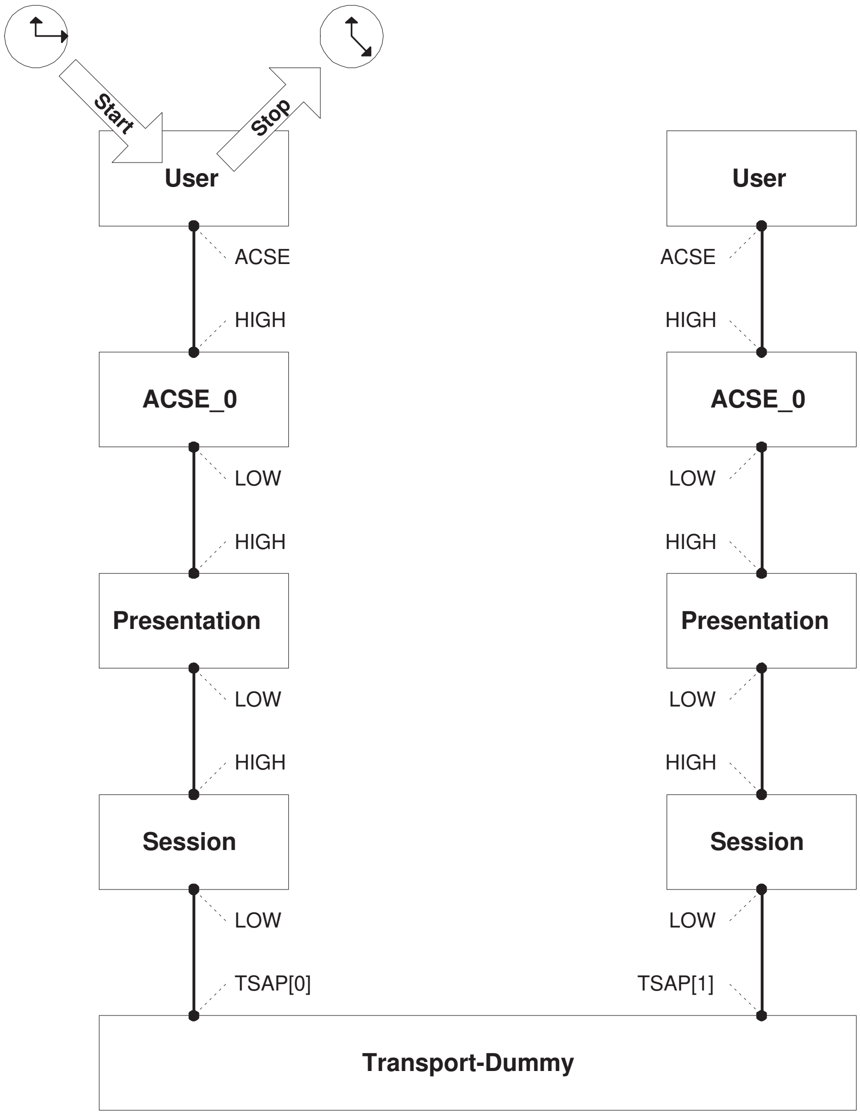

Abbildung 12: Meßaufbau für Protokollstack

Für Informationen zu den Modulen für die Darstellungs– und Sessionschicht sei auf [3] verwiesen.

4.2 Verfahren und Gleichungen

Das Verfahren, mit dem die Zeiten für die einzelnen Varianten und ihre Umgebung ermittelt werden, entspricht folgendem Ablauf:

1. Lies aus Konfigurationsdatei („messung.cnf") die Anzahl der Messungen („nMessVal"), die Anzahl von Verbindungen pro Messung („nAssVal") und Name der Zieldatei für Meßwerte („szNameDat")
2. Starte die Messung und speichere die aktuelle Zeit in der Variablen „startzeit".
3. Baue „nAssVal" Verbindungen auf und wieder ab.
4. Stoppe die aktuelle Messung. Aus der Differenz zwischen aktueller Zeit und „startzeit" ergibt sich die Zeit für „nAssVal" Verbindungen. Speichere diese Differenz in einem Array.
5. Gehe nach 2 bis „nMessVal" Messungen durchgeführt sind.
6. Verwerfe den Wert für die erste Messung[23] und berechne aus den verbliebenen Werten den Mittelwert, Varianz, und die standard Abweichung.
7. Schreibe alle Werte in die durch „szNameDat" angegebene Datei.

Für die Bestimmung der aktuellen Zeit wird die Systemfunktion „clock()" von Unix benutzt. Diese ermöglicht es, die tatsächliche Rechenzeit des Prozesses zwischen zwei Aufrufen zu bestimmen. Dadurch ist die gemessene Zeit unabhängig von anderen Prozessen und sonstigen Dingen, die das Betriebssystem zu erledigen hat.

Damit die „clock()"–Funktion benutzt werden kann, muß das zu spezifizierende System als ein einziger Unix–Prozeß realisiert werden. Dies wird von dem zur Verfügung gestellten Werkzeug zur Übersetzung der Spezifikation in ein lauffähiges Programm (siehe Abschnitt 4.4) nur gewährleistet, wenn das übergeordnete Spezifikationsmodul mit „Systemprocess" und die einzelnen Module mit „Process" attributiert sind. Eine Folge davon ist, daß ein zusätzlicher Aufwand zur Synchronisation der mit „Process" attributierten Module durch das Laufzeitsystem des Werkzeuges erforderlich wird, obwohl die einzelnen Module ohne Einschränkung vollständig asynchron betrieben werden könnten.

[23] Damit wird gewährleistet, daß alle benötigten Resourcen durch das Betriebssystem bereitgestellt sind. Die Ergebnisse in 4.3 rechtfertigen diese Annahme.

Die Gleichungen für die Berechnung der Umgebung sind:

$$2U + P^{'} = t_1 \qquad \text{und} \qquad 2U^{'} + P = t_2 \tag{1}$$

mit

U	Zeit für User–Modul
P	Zeit für Presentation–Modul
$U^{'}$	Zeit für angepaßtes User–Modul
$P^{'}$	Zeit für angepaßtes Presentation–Modul
t_i	Gemessene Zeit für $i = 1, 2$

Da die Struktur und Ablauf der beiden Module User und Presentation–Dummy sich nicht ändert, sondern die Änderungen die zu übertragenden Nachrichtentypen betrifft, d.h. deren Größe, gilt mit den Gleichungen (1) folgende Abschätzung für die Umgebung:

$$t_1 = 2U + P^{'} \quad \leq \quad 2U + P \quad \leq \quad 2U^{'} + P = t_2 \tag{2}$$

Die Begründung ist, daß für die Messung von $2U$ das Presentation–Modul kleinere Datenpakete und für die Messung von P das User–Modul größere Datenpakete bearbeiten muß.

Für die Messung der einzelnen Varianten gilt:

$$\begin{aligned} 2U + 2A + P &= t_3 \\ 2A &= (t_3 - (2U + P)) \end{aligned}$$

mit

U	Zeit für User–Modul
A	Zeit für ACSE–Modul
P	Zeit für Presentation–Modul
t_3	Gemessene Zeit

Mit der Abschätzung (2) ergibt sich für einen erfolgreichen Verbindungsauf– und Verbindungsabbau[24] die Abschätzung:

$$\boxed{t_3 - t_2 \quad \leq \quad 2A \quad \leq \quad t_3 - t_1} \tag{3}$$

[24]Dies entspricht der Zeit $2A$

Die Beziehungen für die Messung des „Speedups“ des verschmolzenen Protokollstacks mit Variante 5 (siehe Abb. 11) gegenüber des Protokollstacks mit Variante 0 (siehe Abb. 12) sind:

$$\begin{array}{llllll} 2U & +2A+2P+2S & +T & = & t_s & \text{für den Protokollstack} \\ 2U & +2V & +T & = & t_v & \text{für die verschmolzene Version} \end{array} \tag{4}$$

mit

U	Zeit für User–Modul
A	Zeit für ACSE–Modul
P	Zeit für Presentation–Modul
S	Zeit für Session–Modul
T	Zeit für Transport–Modul
V	Zeit für die verschmolzenen Module
t_s	Gemessene Zeit für den Protokollstack
t_v	Gemessene Zeit für verschmolzene Version

Es ergibt sich für den Speedup:

$$\boxed{Speedup = \frac{t_s}{t_v} = \frac{2U + 2A + 2P + 2S + T}{2U + 2V + T}} \tag{5}$$

Man beachte, daß in die Berechnung des Speedups die Umgebung $(2U + T)$ mit einfließt. Dies ist deshalb geschehen, da auf Grund der eingeschränkten Meßmöglichkeiten, eine Messung der Umgebung nicht einfach und genau erfolgen kann, wie bei der Messung der Laufzeit für die einzelnen Varianten. Eine Anpassung zum Beispiel der User–Module an das Transport–Modul wäre mit erheblichen Änderungen am User–Modul verbunden. Entsprechendes gilt natürlich für die Anpassungdes Transport–Moduls an die User–Module.

Für den zu messenden Protokollstack bzw. für die verschmolzene Version des Protokollstacks kann man aber sagen, daß der prozentuale Anteil der Umgebung am gesamten System gering ist und damit der Einfluß auf die Laufzeit.

4.3 Ergebnisse

In Tabelle 5 sind die gemessenen Zeiten für die Umgebung und in Tabelle 6 sind die Werte für die Varianten aufgeführt. Die Werte sind in Millisekunden angegeben. Es wurden 10 Messungen *á* 1000 Verbindungen durchgeführt.

Man erkennt, daß die einzelnen Varianten bis auf Variante 4 zwischen 25 msec und 28 msec für die Ausführung eines Verbindungsauf– und wieder Abbaus benötigen, d.h. die einzelnen Varianten unterscheiden sich in ihrem Laufzeitverhalten nicht wesentlich. Grund für dieses Verhalten ist die Umsetzung des Estelle–Codes in C++ –Code durch Pet/Dingo (siehe 4.4). Dieser erzeugte C++ –Code, legt eine strenge Auswertungsreihenfolge der einzelnen Schaltbedingungen fest,

	$2U + P'$	$2U' + P$	Variante
Mittelwert	14,942	15,655	1 bis 3, 5
Varianz	0,002	0,000	
Streuung	0,046	0,014	
Mittelwert	52,802	53,677	4
Varianz	0,001	0,031	
std. Abweichung	0,032	0,176	

Tabelle 5: Gemessene Zeiten für die Umgebung in Millisekunden

Variante	$2U + 2A + P$			$\leq 2A \leq$	
	Mittelwert	Varianz	std. Abweichung		
0	40,850	0,220	0,469	25.195	25,908
1	40,615	0,031	0,177	24.960	25,673
2	41,174	0,129	0,359	25.519	26.232
3	40,939	0,105	0,323	25.284	25.997
4	113,960	0,252	0,502	60.283	61.158
5	42,874	0,542	0,736	27.219	27.932

Tabelle 6: Gemessene Zeiten für die einzelnen Varianten in Millisekunden

d.h. die „From"–Klauseln werden immer vor den „When"–Klauseln ausgewertet[25]. Dies bedeutet, daß Pet/Dingo einen nachrichtenorientierten Stil intern zu einem zustandsorientierten Stil umwandelt.

Variante 4 liegt deutlich außerhalb des Zeitbereiches der anderen Varianten, da die zu übertragenden Pakete durch das Unifizieren die maximal mögliche Größe aller alten Nachrichtentypen besitzen.

Eine Erklärung dafür, daß Variante 5 schlechter abschneidet als die ersten vier Varianten, liegt an den zusätzlichen Prozeduraufrufen im prozeduralen Stil. Die Leistungsfähigkeit von Variante 5 gegenüber den anderen Varianten spielt erst eine Rolle, wenn mehrere Estelle–Module verschmolzen werden. Die gemessenen Werte für einen Protokollstack mit Variante 0 und der verschmolzenen Version mit Variante 5m sind in Tabelle 7 aufgeführt.

Der Vorteil der verschmolzenen Version gegenüber dem ursprünglichen Protokollstack ist deutlich sichtbar. Der errechnete Speedup beträgt $\frac{181{,}667}{50{,}478} \approx 3,6$. Der Unterschied zu dem gemessenen Speedup von 4,5 (siehe [3]) liegt in einem unterschiedlichen Meßaufbau begründet.

Insgesamt kann man sagen, daß die Varianten 0 bis 3 den gleichen Zeitbedarf für eine Verbindung benötigen, da Pet/Dingo, wie bereits erwähnt, einen nachrichtenorientierten Stil in

[25]Für Variante 3 werden die nicht spezifizierten „From"–Klauseln durch eine „if ok = 1"–Abfrage in C++ ersetzt !

mit Variante	Mittelwert	Varianz	std. Abweichung
0	181,667	8,327	2,886
5m	50,478	2,612	1,616

Tabelle 7: Zeiten des Protokollstacks und der verschmolzenen Version

einen zustandsorientierten Stil umwandelt, daß der Stil mit unifizierten Nachrichten (Variante 4) schlecht geeignet ist für die Spezifikation des ACSE–Protokolles, und daß der prozedurale Stil (Variante 5) im Hinblick auf Effizienz am Besten geeignet ist. Der Grund für die Steigerung der Effizienz von Variante 5 liegt darin, daß die Kommunikation zwischen den verschmolzenen Modulen und die Auswahl von Transitionen in den ursprünglichen Modulen wegfallen.

4.4 Systemumgebung und Meßwerkzeuge

Als Meßwerkzeug dient die in Abschnitt 4.2 angegebene „clock()“–Funktion des Unix–Systems. Sie besitzt eine Auflösung von 16,667 msec. Dies hat zur Folge, daß die Messungen entsprechend lange ausgedehnt werden müssen, damit die Zeitdifferenzen zwischen zwei Aufrufen möglichst im Sekundenbereich liegen. Das Meßverfahren gewährleistet dies durch entsprechend viele Verbindungen zwischen zwei Aufrufen. Statistisch läßt sich dann über die Länge der gemessenen Zeit[26] die benötigte Genauigkeit begründen.

Zur Erstellung der ablauffähigen Programme diente die Pet/Dingo–Version 1.0 von Nist (siehe [9], [10]) und der GNU–C++ –Compiler Version 1.40. Pet übersetzt dabei den Estelle–Code in ein Zwischenformat, das von Dingo in C++ –Code umgewandelt wird. Dieser wird mittels des GNU–C++ –Compilers in ein lauffähiges Programm übersetzt.

Es sei an dieser Stelle darauf hingewiesen, daß ein ablauffähiges Programm kontinuierlich Speicher anfordert[27], wenn es mittels „Set“ oder „Packed Array“ definierte Datentypen benutzt. Dies hat erhebliche Auswirkungen auf die Effizienz der einzelnen Varianten[28].

Der für die Messungen benutzte Rechner, ist eine SUN–SPARCstation 10 mit dem Betriebssystem SunOS Version 4.13.

[26]Entspricht der Anzahl von Verbindungen
[27]Dies führt dazu, daß relativ häufig Seiten des Programms ausgelagert werden !
[28]Die gemessenen Werte differierten um einige 10 msec pro Verbindung!

5 Zusammenfassung

Ein Ergebnis dieser Arbeit ist, daß ein nachrichtenorientierter bzw. ein zustandsorientierter Spezifikationsstil in Bezug auf die Lesbarkeit die besten Resultate liefern. Als Begründung seien die unter Abschnitt 3.1 gemachte Definition zur Präzisierung der Lesbarkeit und die im gleichen Abschnitt aufgestellten allgemeinen Regeln für eine Estelle–Spezifikation genannt. Die anderen Stile erfüllen diese Regeln und Konventionen weniger gut und sind deshalb in Bezug auf die Lesbarkeit als schlechter einzustufen. In wieweit diese Regeln eine allgemeingültiges Kriterium zur Überprüfung einer Estelle–Spezifikation darstellen, sei dem Leser überlassen.

Eine Richtschnur, für die Entscheidung welcher der beiden Stilrichtungen, nachrichtenorientiert oder zustandsorientiert, im Sinne der Lesbarkeit bevorzugt werden sollte, ist in Tabelle 1 im Abschnitt 3.3 aufgeführt.

Daß die Lesbarkeit der einzelnen Stile sich verbessern läßt, kann leicht durch die in den einzelnen Abschnitten über die Varianten (siehe Abschnitt 3.3.1 bis 3.3.6) gemachten Vorschläge zu Erweiterungen von Estelle nachgewiesen werden. Insbesondere seien hier die „Interaction–Sets" von R. L. Tenney ([11]) und die von uns vorgeschlagene Erweiterung der „From"–Klausel genannt. Diese stellen eine elegante Möglichkeit dar, Empfangsfehler zu behandeln.

In Bezug auf die Effizienz der einzelnen Stile läßt sich sagen, daß der prozedurale Stil am Besten abschneidet, während die anderen Stilrichtungen (Variante 0 bis 3) prinzipiell nicht untereinander verglichen werden können. Dies ist darauf zurück zuführen, daß das benutzte Werkzeug zur Übersetzung des Estelle–Codes nach C++ –Code eine feste Reihenfolge bei der Auswertung der Schaltbedingungen einführt. Daß der nachrichtenorientierte Stil mit unifizierten Nachrichten eine Verschlechterung gegenüber den anderen Stilen darstellt, ist wesentlich durch die Größe der unifizierten Nachrichten bedingt (siehe dazu auch Abschnitt 3.3.5).

Der gemessene Speedup für das Verschmelzen eines größeren Protokollstacks (Schicht 5 bis 7 im OSI–Basisreferenzmodell) beträgt 3,6 und zeigt deutlich den Vorteil des prozeduralen Stils in Bezug auf die Laufzeiteffizienz gegenüber einer unverschmolzenen Version des Protokollstackes.

Zu dem ACSE–Protokollautomaten ist abschließend zu sagen, daß der Dienstnutzer einen unnötigen Spezialfall beim Verbindungsabbau behandeln muß. Das Problem liegt darin, daß eine Kollision, d.h. beide Dienstnutzer haben gleichzeitig einen Verbindungsabbauwunsch geäußert, auf der Dienstnutzerebene sichtbar wird und entsprechend behandelt werden muß. In Abschnitt 2.3 sind andere, bessere Lösungen für dieses Problem aufgeführt.

6 Literaturverzeichnis

Literatur

[1] Bredereke, J. *Eine Estelle–Erweiterung für strukturiertere Spezifikationen und für einen neuen Spezifikationsstil* interner Bericht (1993)

[2] CCITT Recommendation X.410 *Message Handling Systems: Remote Operation and Reliable Transfer Server* (1984)

[3] Dahl, S. *Spezifikation und Bewertung eines effizienten Kommunikationsprotokolls* Projektarbeit, Univ. Kaiserslautern, FB Informatik (Feb. 1995)

[4] Gotzhein, R. *Vorlesung: Spezifikation von Kommunikationssystemen* Folienkopien, Univ. Kaiserslautern, FB Informatik (SS 1994)

[5] Hogrefe, D. *Estelle, LOTOS und SDL. Standard–Spezifiktionssprachen für verteilte Systeme* Springer (1989)

[6] ISO/TC 97, ISO 8649 *Information Processing Systems — Open Systems Interconnection — Service definition for the Association Control Service Element* (1988)

[7] ISO/TC 97, ISO 8650 *Information Processing Systems — Open Systems Interconnection — Protocol specification for the Association Control Service Element* (1988)

[8] ISO/TC 97/SC 21, ISO 9074 *Information Processing Systems — Open Systems Interconnection — Estelle: A Formal Description Technique Based on an Extended State Transition Model* (1989)

[9] Sijelmassi, R. und Strausser, B. *The Portable Estelle Translator: an overview and user guide* Tech. Rep. NCSL/SNA—91/2, U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD (Jan. 1991)

[10] Sijelmassi, R. und Strausser, B. *The Distributed Implementation Generator: an overview and user guide* Tech. Rep. NCSL/SNA—91/3, U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD (Jan. 1991)

[11] Tenney, Richard L. *Adding Interaction Sets to Estelle* In: Juan Quemada and Jose Maños and Enrique Vazquez: 3rd International Conference on Formal Description Techniques for Distributed Systems – FORTE'90, Madrid, North–Holland (5.-8. Nov. 1990)

[12] Turner, K. *Using Formal Description Techniques — An Introduction to Estelle, LOTOS and SDL* Wiley (1992)

A Bezeichnungen

A.1 Interaktionspunkte

Bezeichnungen für die Interaktionspunkte (siehe Sturkturbild 1 in Kapitel 2):

Modul	**IP**	**Verbunden mit**	
		Modul	**IP**
User_*	ACSE	ACSE_*	HIGH
ACSE_*	HIGH	User_*	ACSE
ACSE_*	LOW	PRES_*	HIGH
PRES_*	HIGH	ACSE_*	LOW

A.2 Zustände

In folgender Tabelle sind die Bezeichnungen und Bedeutungen der Zustände im Spezifikationstext bzw. in der Norm aufgelistet. Für weitere Informationen sei auf Kapitel 2 bzw. auf die Normen 8649 (siehe [6]) und 8650 (siehe [7]) verwiesen.

Spezifikaion	**Norm**	**Bedeutung**
IDLE	Idle–Unassoc	Verbindung ist nicht aufgebaut, Ruhezustand
AWAIT_AARE	Awaiting AARE	PDU mit Bestätigung oder Ablehnung des Aufbauwunsches wird von Gegenüber erwartet
AWAIT_ASSRP	Awaiting A–ASCrsp	Entscheidung des Nutzers über Aufbauwunsch des Partners wird erwartet
ASSOCIATED	Associated	Verbindung ist aufgebaut, Datenübertragung kann stattfinden
AWAIT_RLRE	Awaiting RLRE	Warten auf Entscheidung des Partners bzgl. des Abbauwunsches
AWAIT_RLRP	Awaiting A–RLSrsp	PDU mit Entscheidung über den Abbauwunsch des Partners wird vom Nutzer erwartet
COLLISION_INITIATOR	Collision Association Initiator	Dient zur Behandlung von Kollisionen. Der Initiator der Verbindung wechselt in diesen Zustand
COLLISION_RESPONDER	Collision Association Responder	Dient zur Behandlung von Kollisionen. Der „Responder" der Verbindung wechselt in diesen Zustand

A.3 Nachrichten

Bezeichnungen für die Nachrichten von Variante 1 bis Variante 3:

Spezifikation	**Norm**	**I/O**	**Bedeutung**
A_ASSreq	A–ASCreq	I	ACSE Association Request Aufbauwunsch äußern
A_ASSind	A–ASCind	O	ACSE Association Indication Anzeigen eines Aufbauwunsches
A_ASSrsp	A–ASCrsp	I	ACSE Association Response Aufbauwunsch wird bestätigt oder abgelehnt
A_ASScnf	A–ASCcnf	O	ACSE Association Confirm Entscheidung über Aufbauwunsch wird angezeigt
A_RELreq	A–RLSreq	I	ACSE Release Request Abbauwunsch äußern
A_RELind	A–RLSind	O	ACSE Release Indication Abbauwunsch wird dem Nutzer mitgeteilt
A_RELrsp	A–RLSrsp	I	ACSE Release Response Abbauwunsch wird bestätigt oder abgelehnt
A_RELcnf	A–RLScnf	O	ACSE Release Confirm Entscheidung über Abbauwunsch wird angezeigt
A_ABreq	A–ABRreq	I	ACSE Abort Request Abbruch äußern
A_ABind	A–ABRind	O	ACSE Abort Indication Anzeigen eines Abbruchs
A_PABind	A–PABind	O	ACSE Provider Abort Indication Anzeigen eines Abbruchs durch Darstellungsschicht
P_CONreq	—	O	Presentation Connection Request
P_CONind	—	I	Presentation Connection Indication
P_CONrsp	—	O	Presentation Connection Response
P_CONcnf	—	I	Presentation Connection Confirm
P_RELreq	—	O	Presentation Release Request
P_RELind	—	I	Presentation Release Indication
P_RELrsp	—	O	Presentation Release Response
P_RELcnf	—	I	Presentation Release Confirm
P_UABreq	—	O	Presentation User Abort Request
P_UABind	—	I	Presentation User Abort Indication
P_PABind	—	I	Presentation Provider Abort Indication

I: Vom ACSE–Modul empfangene Nachrichten (Input)

O: Vom ACSE–Modul gesendete Nachrichten (Output)

Bezeichnungen für die Nachrichten von Variante 4:

Nachrichten	**I/O**	**Bedeutung**
A_MSGreq	I	ACSE Message Request Enthält alle ursprünglich empfangbare Nachrichten
P_MSGind	O	ACSE Message Request Enthält alle ursprünglich sendbaren Nachrichten

I : Vom ACSE–Modul empfangene Nachricht (Input)

O: Vom ACSE–Modul gesendete Nachricht (Output)

Bezeichnungen für die Prozeduren und Nachrichten von Variante 5:

Prozeduren	**I/O**	**Bedeutung**
ACSE_Init	I	ACSE Initialize Initialisierungsroutine
ACSE_A_ASSreq	I	ACSE Association Request
ACSE_A_ASSrsp	I	ACSE Association Response
ACSE_A_RELreq	I	ACSE Release Request
ACSE_A_RELrsp	I	ACSE Release Response
ACSE_A_ABreq	I	ACSE Abort Request
ACSE_P_CONind	I	ACSE Presentation Connection Indication
ACSE_P_CONcnf	I	ACSE Presentation Connection Confirm
ACSE_P_RELind	I	ACSE Presentation Release Indication
ACSE_P_RELcnf	I	ACSE Presentation Release Confirm
ACSE_P_UABind	I	ACSE Presentation User Abort Indication
ACSE_P_PABind	I	ACSE Presentation Provider Abort Indication
Pin_PCONreq	O	Presentation Connection Request
Pin_PCONrsp	O	Presentation Connection Response
Pin_PRELreq	O	Presentation Release Request
Pin_PRELrsp	O	Presentation Release Response
Pin_PUABreq	O	Presentation User Abort Request

I : Prozeduren für empfangene Nachrichten (Input)

O: Prozeduren für gesendete Nachrichten (Output)

Nachrichten von und zum Dienstnutzer:

Nachrichten	**I/O**	**Bedeutung**
A_ASSreq	I	ACSE Association Request
A_ASSind	O	ACSE Association Indication
A_ASSrsp	I	ACSE Association Response
A_ASScnf	O	ACSE Association Confirm
A_RELreq	I	ACSE Release Request
A_RELind	O	ACSE Release Indication
A_RELrsp	I	ACSE Release Response
A_RELcnf	O	ACSE Release Confirm
A_ABreq	I	ACSE Abort Request
A_ABind	O	ACSE Abort Indication
A_PABind	O	ACSE Provider Abort Indication

I : Vom ACSE–Modul empfangene Nachrichten (Input)

O: Vom ACSE–Modul gesendete Nachrichten (Output)

B Quelltexte und Dateien

Die Quelltexte für die einzelnen Varianten sind in den Dateien

Type.e:	Enthält die Typdefinitionen
Channel.e:	Kanaldefinition für Variante 0 bis Variante 3 und Variante 5
Channel_4.e:	Kanaldefinition für Variante 4
Proc.e:	Die Prozeduren für die Transitionsblöcke. (Ab Variante 1)
ACSE_0.e:	Variante 0
ACSE_1.e:	Variante 1
ACSE_2.e:	Variante 2
ACSE_3.e:	Variante 3
ACSE_4.e:	Variante 4
ACSE_5.e:	Variante 5
ACSE_5m.e:	Variante 5m zur Verschmelzung mit Schicht 6 und 5
ACSE_5.for:	„Forward“ Deklarationen der Prozeduren von Variante 5
Addtype.e:	Enthält zusätzliche Datentypen für Variante 5

Für die Laufzeitmessung werden die Dateien

m.stl:	Spezifikationsmodul mit Köpfen und Verbindungsstruktur
User.e:	Enthält einen Nutzer als Initiator und einen als „Responder“
User_p.e:	An „Pres.e“ angepaßter User
Pres.e:	„Presentation–Layer“ als Verbindungsschicht zwischen den zwei ACSE–Modulen für Variante 0 bis Variante 3 und Variante 5
Pres_u.e:	An „User.e“ angepaßtes Verbindungsmodul
User_4.e:	User für Variante 4
User_4p.e:	An „Pres_4.e“ angepaßter User
Pres_4.e:	„Presentation–Layer“ für Variante 4
Pres_4u.e:	An „User_4.e“ angepaßtes Verbindungsmodul
Protokoll.e:	Übernimmt die Kommunikation zum User– bzw. Transport–Modul in dem verschmolzenen Protokollstack mit Variante 5m
messung.cc:	Enthält die C++ –Funktionen zum Messen
messung.cnf:	Konfigurationsdatei für „messung.cc“

und es werden die Dateien (siehe [3]) der Schichten 6 bis 4 des OSI–Referenzmodells für die Messung des Speedups des prozeduralen Stils eines verschmolzenen Protokollstacks mit Variante 5m gegenüber der unverschmolzenen Version des Protokollstacks mit Variante 0 (siehe Kapitel 4) benötigt.