Formale Entwicklung verteilter reaktiver Systeme mit FOCUS*

Max Breitling, Ursula Hinkel, Katharina Spies

Institut für Informatik, Technische Universität München

e-mail:(breitlin|hinkel|spiesk)@informatik.tu-muenchen.de

1 Einleitung

Formale Methoden und Techniken liefern aufgrund ihrer mathematisch-logischen Fundierung die Voraussetzung zur Entwicklung komplexer Systeme, an die hohe Sicherheits- und Qualitätsanforderungen gestellt werden. Nur eine formal abgesicherte Basis macht es möglich, präzise Spezifikationen zu erstellen, den formalen Nachweis der Gültigkeit von Systemeigenschaften zu erbringen sowie eine Systementwicklung in mehreren Schritten durchzuführen, wobei die Spezifikationen zunehmend detaillierter werden und zueinander in einer beweisbaren Verfeinerungsbeziehung stehen. FOCUS ist eine an der TU München am Lehrstuhl von Prof. M. Broy entwickelte Methodik zur formalen Entwicklung verteilter, reaktiver Systeme, die diese Zielsetzung verfolgt. Damit FOCUS in Zukunft zur vollständigen Entwicklung industrieller Softwareprodukte für technische Systeme eingesetzt werden kann, zielen aktuelle Arbeiten verstärkt darauf ab, die Praxistauglichkeit von FOCUS durch die Behandlung komplexer Aufgabenstellungen zu überprüfen und nachzuweisen.

Mit dem vorliegenden Papier wird FOCUS als formale Entwicklungsmethodik vorgestellt. Die verschiedenen Beschreibungstechniken von FOCUS sowie das Verfeinerungskonzept werden ausgehend von der einheitlichen semantischen Basis skizziert. Der Schwerpunkt liegt jedoch in der Beschreibung von drei bereits durchgeführten komplexen Anwendungen: Die Modellierung von Betriebssystemkonzepten, die Spezifikation und Verifikation eines TIMEWARP-Simulators und die semantische Fundierung von SDL in Verbindung mit einer Verifikationsmethode.

2 Die Methodik FOCUS

FOCUS dient zur formalen Modellierung und Entwicklung verteilter, reaktiver Systeme auf der Basis präziser, formaler Grundlagen. Ein verteiltes System wird immer als Netzwerk interagierender Komponenten modelliert, deren Zusammenspiel durch den Austausch von Nachrichten

*Diese Arbeit wurde unterstützt durch den SFB 342 der DFG "Werkzeuge und Methoden für die Nutzung paralleler Rechnersysteme" und durch die Bayerische Forschungsstiftung im Rahmen des Projekts FORSOFT.

über gerichtete Verbindungskanäle erfolgt. Dieses Kapitel gibt eine kurze Einführung in die semantischen Grundlagen, die Beschreibungstechniken und das Verfeinerungskonzept von FOCUS. Ausführlichere Beschreibungen und Literatur finden sich in [1].

Aufgrund seiner allgemeinen Ausrichtung wird FOCUS auf verschiedene Art und Weise verwendet: Der formale Rahmen sowie die zugehörigen Beschreibungstechniken werden zur reinen *Spezifikationserstellung* und *Modellierung* eingesetzt. Basierend auf den dabei erstellten Formalisierungen können *Eigenschaftsbeweise* geführt werden. Der zusätzliche Einsatz des Verfeinerungskonzepts ermöglicht vollständige *Top-Down-Entwicklungen* von Systemen. Dabei wird zunächst eine einfache, abstrakte Spezifikation erstellt, die die Basis für die schrittweise Präzisierung der Systembeschreibung in Richtung der letztendlich zu erstellenden Implementierung bildet. Die Entwicklungsschritte sind durch Entwurfsentscheidungen und die Strukturierung in Unterkomponenten charakterisiert. Die Korrektheit solcher Verfeinerungsschritte wird mittels vollständig geführter formaler Beweise nachgewiesen. Weiterhin können *semantische Fundierungen* für Programmiersprachen oder industriell verbreitete Beschreibungstechniken erstellt werden, für die keine oder eine nur ungenügende formale Basis vorliegt.

2.1 Semantische Grundlagen

Der Nachrichtenaustausch zwischen den Komponenten eines Netzwerks wird durch gezeitete *Nachrichtenströme* modelliert. Ein Strom enthält alle Nachrichten, die über einen Kanal gesendet werden, sowie Information darüber, in welchem Zeitintervall eine Nachricht übertragen wird. Dazu wird von einer globalen, diskreten Systemzeit ausgegangen. Das Voranschreiten der Zeit wird durch das Einfügen von speziellen Symbolen $\surd$, sogenannten Zeitticks, in den Strömen dargestellt. So beginnt der Strom

$$a \surd ab \surd \surd bca \surd \ldots$$

mit den Nachrichten *aabbca*. Im ersten Zeitintervall wird a, im zweiten a gefolgt von b übertragen; im dritten Zeitintervall findet keine Nachrichtenübertragung statt, im vierten werden die Nachrichten b, c, a übertragen. Eine vollständige Kommunikationsgeschichte eines Kanals wird durch einen Strom mit unendlich vielen Zeitticks modelliert. $N^{\underline{\infty}}$ bezeichnet die Menge der gezeiteten Nachrichtenströme über der Nachrichtenmenge N.

Das Verhalten einer Komponente wird durch ihr Ein-/Ausgabeverhalten mittels *stromverarbeitender Funktionen* spezifiziert. Diese definieren die Beziehung zwischen den Strömen, die die Komponente auf ihren Eingabekanälen erhält, und den Strömen, die sie auf ihren Ausgabekanälen sendet. Für die Komposition von Komponenten zu Systemen stehen parallele und sequentielle Komposition sowie Rückkopplung zur Verfügung. Das Verhalten eines zusammengesetzten Systems ergibt sich eindeutig aus dem Verhalten seiner Komponenten.

Sei S die Spezifikation einer Komponente, die Nachrichten der Menge I erhält und Nachrichten der Menge O sendet. Mit der Spezifikation wird eine Relation R_S über Ein- und Ausgabeströmen beschrieben. Die Semantik $[\![S]\!]$ der Komponente ist die Menge aller stromverarbeitenden Funktionen, für die gilt:

$$[\![\ S\ ]\!] =_{\text{def}} \{f_S : I^{\underline{\infty}} \to O^{\underline{\infty}} \mid f_S \text{ ist stromverarbeitend } \wedge\ \forall i \in I^{\underline{\infty}} : R_S[i; f_S(i)]\}$$

Dabei erfüllen stromverarbeitende Funktionen nach Definition weitere, hier nicht genannte semantische Eigenschaften, um ausführbares Verhalten zu beschreiben. So darf beispielsweise die aktuelle Ausgabe einer Komponente nicht durch zukünftige Eingaben beeinflußt werden. Bezüglich des zeitlichen Verhaltens einer Komponente werden folgende Spezifikationsformate unterschieden:

- *Gezeitete* Spezifikationen beschreiben das zeitliche Verhalten einer Komponente explizit.
- Bei *ungezeiteten* Spezifikationen wird auf Zeitinformation in den Strömen verzichtet – Ströme sind dann endliche oder unendliche Nachrichtensequenzen ohne Zeitticks. Eine ungezeitet spezifizierte Komponente darf sich bezüglich der Zeit beliebig verhalten.
- Die *synchrone* Zeitmodellierung eignet sich für Systeme, deren Komponenten synchron oder getaktet kommunizieren. In diesem Spezifikationsformat werden gezeitete Nachrichtenströme verwendet, die in jedem Zeitintervall höchstens eine Nachricht enthalten.

Eine Erweiterung des semantischen Modells erlaubt die Modellierung mobiler, dynamischer Systeme. Dabei handelt es sich um Systeme, deren Kommunikationsstruktur sich während des Systemablaufs ändert und in denen Komponenten dynamisch kreiert und in die bisherige Systemstruktur integriert werden. Die Menge der Nachrichten wird dazu um *Ports* erweitert. Ein Port ist ein Kanalname mit einem Zugriffsrecht. Durch das Senden bzw. Empfangen von Ports ändert sich die Menge der verfügbaren Ein- und Ausgabekanäle einer Komponente. In diesem Modell wird das Verhalten einer Komponente durch mobile Funktionen beschrieben. Diese bilden wie stromverarbeitende Funktionen Ein- auf Ausgabeströme ab und erfüllen dabei zusätzlich die Eigenschaft der *Vertraulichkeit*. Diese stellt sicher, daß die Komponente nur auf Kanäle zugreift, zu denen sie aktuell die Zugriffsrechte besitzt.

2.2 Beschreibungstechniken

Zur geeigneten Formalisierung der Systemeigenschaften stellt FOCUS eine Reihe von mathematisch-textuellen und graphischen Beschreibungstechniken bereit. Diese verfügen jeweils über eine wohldefinierte, mathematisch-logisch formalisierte Semantik, wodurch präzise festgelegt wird, welche Bedeutung eine formale Systembeschreibung in FOCUS hat. Die Auswahl und Festlegung der Beschreibungstechniken ermöglicht die Spezifikation eines Systems aus verschiedenen Sichten, für die jeweils zugeschnittene Beschreibungstechniken vorliegen.

Zur textuellen und an mathematischen Notationen orientierten Formulierung von Spezifikationen können in Kombination mit speziellen Operatoren sowohl ein *relationaler Stil* als auch *Funktionsgleichungen* verwendet werden. Der relationale Stil ist sehr eng an die semantische Basis angelehnt, während in Gleichungen bereits das schrittweise Abarbeiten eines Nachrichtenstroms zum Ausdruck kommt; ein Beispiel wird in Abschnitt 3.3 gezeigt. Für Anwender, die im Umgang mit Formeln weniger geschult sind, werden die im folgenden skizzierten graphischen Beschreibungstechniken angeboten. Durch den Einsatz dieser Techniken können Spezifikationen erstellt werden, ohne mathematische Formalisierungen *explizit* verwenden zu müssen.

Die Spezifikation der Struktur von Netzwerken und der Schnittstelle von Komponenten erfolgt in FOCUS durch Datenflußdiagramme, sogenannten *Systemstrukturdiagrammen* (SSD, Abb.

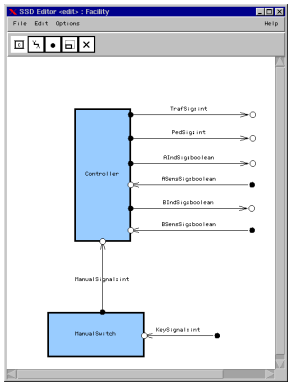

Abb. 1.1: SSD

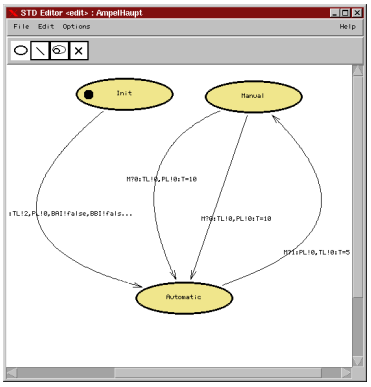

Abb. 1.2: STD

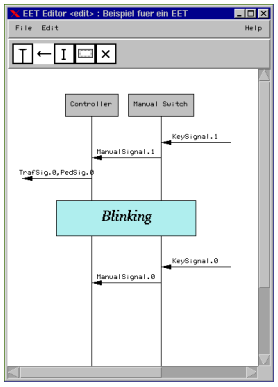

Abb. 1.3: EET

Abbildung 1: Beschreibungstechniken

1.1). Diese können hierarchisch aufgebaut sein, d.h. einer Komponente kann ihrerseits ein SSD zugeordnet sein, das die Substruktur der Komponente beschreibt; ein Beispiel für ein komplexes SSD zeigt Abbildung 3. SSDs können direkt in eine Darstellung in ANDL (Agent Network Description Language) umgesetzt werden. Damit erfolgt die Festlegung von Schnittstelle und Struktur in einer schematisierten programmiersprachenähnlichen Notation. Zudem kann so eine maschinengerechte Darstellung der Spezifikationen von Komponenten erreicht werden, die eine Anbindung von Verifikationswerkzeugen ermöglicht. Mit *Datentypdefinitionen* (DTDs) werden Datentypen definiert, die für die Definition von Kanälen, Nachrichten oder Zustandsvariablen benötigt werden. Da in FOCUS der Aspekt der Datenmodellierung konzeptionell nicht im Vordergrund steht, werden DTDs in textueller Form erstellt.

Zur Spezifikation des Verhaltens dienen *Zustandsübergangsdiagramme* (STD, Abb. 1.2), die erweiterten endlichen Automaten entsprechen. Ein Übergang zwischen zwei (Kontroll-) Zuständen wird mit einem Paar beschriftet, das aus einer empfangenen und der als Reaktion gesendeten Nachricht besteht. Ferner können Vor- und Nachbedingungen formuliert werden, die Aussagen über die Änderung des Datenzustandes treffen. Eine weitere Möglichkeit zur Verhaltensspezifikation bietet eine *tabellarischen Form*, eine textuell orientierte Darstellung der STDs; ein Beispiel hierfür zeigt Abbildung 4. Durch *Erweiterte Ereignisdiagramme* (EET, Abb. 1.3), einer Variante von Message Sequence Charts (MSCs), werden Abläufe des Gesamtsystems als Interaktionen zwischen den Systemkomponenten angegeben.

Das prototypische Werkzeug AUTOFOCUS dient der Unterstützung eines Anwenders beim Einsatz von FOCUS. AUTOFOCUS verfügt über die in Abbildung 1 gezeigten Editoren für die Beschreibungstechniken SSD, STD und EET und ermöglicht eine automatische Konsistenzüberprüfung zwischen verschiedenen Dokumenten. Zur Validierung von Spezifikationen steht eine Simulationsumgebung bereit, mit der das Verhalten von Systemen erprobt und animiert werden kann. Ein Anschluß an klassische Theorembeweiser und Model Checker ist in Vorbereitung.

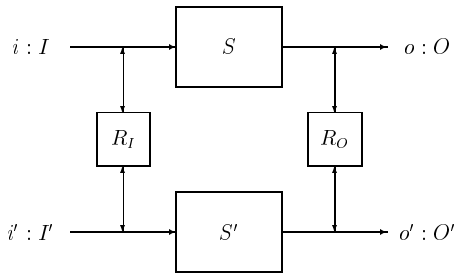

Abbildung 2: Interaktionsverfeinerung

2.3 Verfeinerungen mit FOCUS

FOCUS bietet ein kompositionales Verfeinerungskonzept an, das es ermöglicht, die Verfeinerungsrelation zwischen Spezifikationen verschiedener Abstraktionsebenen formal zu beschreiben. Die zentrale Verfeinerungsrelation von FOCUS ist die *Interaktionsverfeinerung* (IAV). Eine gültige IAV zwischen zwei Komponenten S und S' besagt, daß diese ein im wesentlichen gleiches, aber auf unterschiedlichen Abstraktionsebenen spezifiziertes Verhalten aufweisen. Die Verbindung zwischen den Abstraktionsebenen wird durch Relationen R_I und R_O beschrieben, wie in Abbildung 2 dargestellt. Für jedes Strompaar (i', o'), das ein konkretes Verhalten von S' darstellt, muß es ein passendes abstraktes Strompaar (i, o) geben, das ein Verhalten von S repräsentiert. Dies wird formalisiert durch

$$S \overset{(R_I;R_O)}{\rightsquigarrow} S' \quad \Leftrightarrow_{\text{def}} \quad \forall i', o' : R_{S'}[i'; o'] \;\Rightarrow\; \exists i, o : R_I[i; i'] \wedge R_S[i; o] \wedge R_O[o; o']$$

Erfüllen R_I und R_O gewisse Bedingungen, ist die wichtige Eigenschaft der *Kompositionalität* sichergestellt. Die Verfeinerung der Teilkomponenten eines Systems führt damit ohne weitere Beweisverpflichtungen zu einer Verfeinerung des Gesamtsystems.

Die IAV läßt sich zu konkreten Varianten spezialisieren. Mit R_I und R_O als Identitätsrelationen wird die *Verhaltensverfeinerung* realisiert, mit der das Verhalten einer Komponente durch Entwurfsentscheidungen konkretisiert werden kann. Mit einer *Schnittstellenverfeinerung* ist es möglich, die Schnittstelle einer Komponente durch die Konkretisierung von Anzahl und Typen ihrer Kommunikationskanäle zu verändern. Durch eine *strukturelle Verfeinerung* wird der statische Aufbau des verteilten Systems entwickelt, indem Komponenten zu Netzwerken von interagierenden Komponenten verfeinert werden. Für alle Varianten stehen spezifische Beweisregeln zur Verfügung, mit denen Verfeinerungsrelationen formal bewiesen werden können.

3 FOCUS in der Anwendung

Die Anwendbarkeit formaler Methoden und auch von FOCUS wurde bisher meist an Beispielen gezeigt, die aus theoretischer Sicht interessant sind und die Mächtigkeit der Methoden demonstrieren. Oftmals haben diese Beispiele jedoch aus Anwendersicht nur wenig Bezug zu

realen Problemen und deren Vielschichtigkeit. Da die Festlegung der semantischen Basis von FOCUS weitgehend konsolidiert und abgeschlossen ist, zielen aktuelle und zukünftige Arbeiten vor allem darauf ab, die Praktikabilität von FOCUS zu überprüfen und nachzuweisen.

Um mögliche Einsatzgebiete von FOCUS aufzuzeigen, werden drei Anwendungen vorgestellt. Dabei werden jeweils die Idee und Zielsetzung, deren Behandlung mit FOCUS und abschließend die Ergebnisse beschrieben. Einen Überblick über in FOCUS erstellte Fallstudien gibt [2].

3.1 Formale Modellierung von Betriebssystemkonzepten

Als erste große Anwendung zur *inkrementellen Spezifikationsentwicklung* zeigt [3], daß FOCUS erfolgreich zur Modellierung von Betriebssystemkonzepten eingesetzt werden kann. Diese werden auf hohem Abstraktionsniveau und ohne Einbeziehung von Realisierungsdetails behandelt. Der Schwerpunkt der Arbeit liegt in der systematischen und methodischen Erstellung komplexer Spezifikationen im mathematisch-orientierten Spezifikationsstil. Die Basis bildet die semantische Erweiterung zur Beschreibung mobiler, dynamischer Systeme.

Betriebssysteme sind langlebige Softwareprodukte, die für die Nutzung von Rechensystemen unverzichtbar sind und deren systematische Entwicklung mit formalen Verfahren bisher nur wenig betrachtet wurde. Sie sind ein komplexes Anwendungsgebiet, das eine Herausforderung für den Einsatz von FOCUS darstellt. Die erstellte Modellierung ist durch folgende Sichtweise auf *klassische Betriebssysteme* charakterisiert: Ein Betriebssystem sorgt für die angemessene und faire Nutzung der Ressourcen, wie beispielsweise *Speicher* oder *Prozessoren*, wodurch sich verschiedenste Verwaltungsaufgaben ergeben. Die zentrale Aufgabe besteht somit in der Planung, Organisation und Kontrolle aller Berechnungen, die zur Durchführung von Anwendungen mit einem Rechensystem erforderlich sind.

Zunächst werden die zentralen Anforderungen, die ein klassisches Betriebssystem charakterisieren, anhand der entsprechenden Standardliteratur erarbeitet und festgelegt. Bereits der nächste Schritt, das Erstellen der ersten Formalisierung, stellt im allgemeinen wegen der komplexen und keineswegs leicht verständlichen formalen Konzepte von FOCUS eine schwer zu bewältigende Hürde dar. Wegen seiner Basisfunktion für eine formale Systementwicklung – jeder weitere Entwicklungsschritt baut auf der ersten Formalisierung auf – ist dieser erste Schritt jedoch mit großer Sorgfalt durchzuführen. Um Anwendern Hilfestellung für das Erlernen der speziellen Notationen zu geben, werden Spezifikationsmuster definiert, die eine schematische Erstellung formaler Spezifikationen ausgehend von einer Beschreibung in strukturierter textueller Form ermöglichen. Die Muster werden in der konkreten Anwendung konsequent eingesetzt.

Die methodische Vorgehensweise ist dadurch gekennzeichnet, daß zunächst ein *einfaches* System mit *Kernfunktionalität* modelliert wird, zu dem schrittweise weitere Funktionalitäten hinzugenommen werden. Die Funktionalität orientiert sich durchgängig am bekannten Zustandsdiagramm für Betriebssystemprozesse. Durch die Modellierung wird die Analogie zwischen den Zustandsübergängen und den für die Ressourcenverwaltung erforderlichen Maßnahmen bei der Ausführung einer Berechnung (atomare Ausführung von Instruktionen, Adressenbestimmung, Unterbrechung bei fehlenden Ressourcen) explizit am Verhalten kooperierender Komponenten aufgezeigt. Die inkrementelle Entwicklung des spezifizierten abstrakten Betriebssystems wird

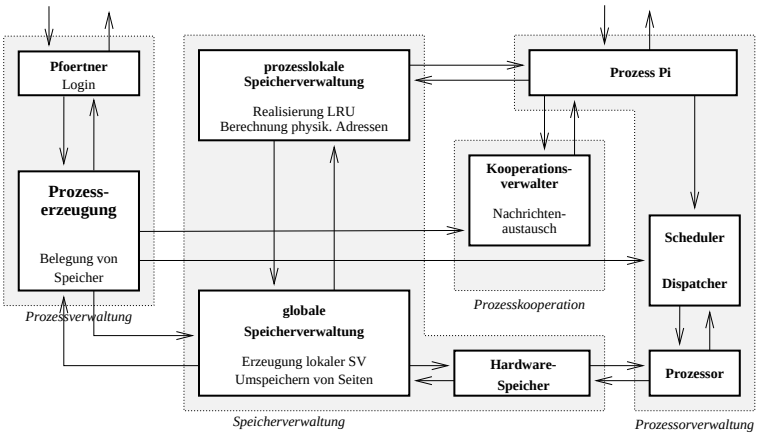

Abbildung 3: Grobstruktur des modellierten Betriebssystems als SSD

in vier Phasen durchgeführt; Abbildung 3 zeigt das vereinfachte verteilte System.

- Das Kernstück bildet die *Prozessorverwaltung* mit Round-Robin-Scheduling, Dispatching und der expliziten Prozessorzuteilung. Somit benötigen Betriebssystemprozesse zunächst ausschließlich die Ressource *Prozessor*, um ihre Berechnung ausführen zu können.
- Der nächste Schritt umfaßt Aufgaben der *Speicherverwaltung* mit virtuellem Speicher, einem prozeßlokalen Seitenersetzungsverfahren gemäß LRU sowie der Berechnung physikalischer Adressen. Alle Erweiterungen des Verhaltens sind dadurch charakterisiert, daß ein Prozeß nun zusätzlich die Ressource *Speicherplatz* benötigt.
- Die dritte Entwicklungsphase behandelt Konzepte der *Prozeßkooperation* mittels Nachrichtenaustausch und den Alternativen *blockierendes Senden* und *nichtblockierendes Senden*. Prozesse werden zu Gruppen zusammengefaßt, deren Mitglieder kooperieren können.
- Abschließend wird ein Subsystem zur *Prozeßerzeugung* mit einer *Login*-Komponente in die Modellierung integriert. Ausgelöst durch die Umgebung werden Prozesse erzeugt, falls genügend Speicherplatz zur Verfügung steht. Einem neuen Prozeß wird eine auszuführende Berechnung zugewiesen, und er wird einer Prozeßgruppe zugeordnet.

Die Stärke der gewählten Vorgehensweise liegt im systematischen Umgang mit den Spezifikationen: Die Weiterentwicklung der Formalisierung orientiert sich an der steigenden Komplexität der Aufgabenstellung. Aufgrund vorbereitender Maßnahmen in *einfachen* Spezifikationen entstehen deren Erweiterungen durch Anpassungen und Vervollständigungen. Die Erstellung der Formalisierungen ist nachgeordnet, und der Entwickler kann sich auf den Inhalt der ihm gestellten Aufgabe konzentrieren. Die Veränderungen an bestehenden Spezifikationen entsprechen den aufgrund der erweiterten Funktionalität zu erwartenden Anpassungen, ohne daß zusätzliche durch FOCUS begründete Maßnahmen zu ergreifen sind. Die formalen Modellierungen der Systeme mit wachsender Komplexität sind schrittweise nachvollziehbar und werden dadurch leichter verständlich als die direkte Modellierung des Systems in seinem vollen Umfang. Zudem lassen sich viele Betriebssystemkonzepte anhand der abstrakten Modellierung gut erklären.

Die Ergebnisse der Arbeit liefern für FOCUS eine Anwendung zur formalen Spezifikationsent-

wicklung, zur Demonstration der Praxistauglichkeit und die Entwicklung methodischer Anleitungen. Die Spezifikation eines Systems, dessen Verhalten an einer komplexen Standardanwendung aus der praktischen Informatik orientiert ist, wird vollständig entwickelt. Zusätzlich wird ein möglicher Weg aufgezeigt, wie Formalisierungen in FOCUS systematisch und schrittweise erstellt werden können.

3.2 Verifikation eines TIMEWARP-Simulators

Die Aufgabe der in diesem Abschnitt beschriebenen Fallstudie war es, die Korrektheit eines verteilten, mit TIMEWARP synchronisierten Simulators unter Verwendung der streng formalen Verfeinerungsrelationen von FOCUS in einer Top-Down-Entwicklung nachzuweisen. Eine ausführliche Beschreibung der Fallstudie mit Literaturangaben findet sich in [4].

Die Synchronisierung des TIMEWARP eignet sich für Simulationsmodelle, deren Abläufe sich adäquat durch Zustandsänderungen zu diskreten Zeitpunkten beschreiben lassen. Diese Zustandsübergänge werden durch Ereignisse beschrieben, die angeben, *wann* sich der Zustand des Modells *wie* ändert. Wird ein Simulator verteilt realisiert, so sind die Komponenten des Simulators, die jeweils einzelne Partitionen des Modells bearbeiten, i.a. nicht unabhängig voneinander, sondern müssen Ereignismengen austauschen. In einem konservativen Ansatz warten die Komponenten mit der Simulation eines Zeitpunktes t solange, bis alle für t relevanten Ereignisse verfügbar sind. Dagegen rechnen im TIMEWARP-Ansatz die Komponenten unter der optimistischen Annahme, alle Ereignisse bereits zu kennen, voraus. Werden dann noch Ereignisse empfangen, die zu berücksichtigen gewesen wären, so muß eine Komponente einen *Rollback* ausführen. Dazu setzt sie auf einen passenden Zustand zurück und storniert alle Ausgaben, die durch den Rollback ungültig geworden sind. In diesem Ansatz wird also keine Zeit mit Warten vergeudet; dafür muß Mehraufwand für den Umgang mit verspäteten Ereignissen und Stornierungen und eine Speicherung von alten Berechnungszuständen geleistet werden. In der Praxis werden bei geeigneter Partitionierung durch Anwendung des TIMEWARP schnellere Antwortzeiten eines verteilten Simulators erreicht.

Die Korrektheit eines derartigen Simulators umfaßt folgende drei Aspekte: Zum einen muß das Simulationsmodell eine adäquate Abstraktion sein, und die eigentlichen Simulationsschritte müssen korrekt berechnet werden. Ferner muß nachgewiesen werden, daß trotz auftretender Stornierungen und Rollbacks ein Fortschritt in der gesamten Berechnung erreicht wird. Schließlich müssen die Komponenten so interagieren, daß ein korrektes endgültiges Simulationsergebnis erreicht wird. Der erste Aspekt ist sehr anwendungsspezifisch, der zweite Aspekt ist in der Literatur bereits genauer untersucht worden. Daher wurde in dieser Fallstudie der dritte Aspekt behandelt, während die ersten beiden als korrekt postuliert werden.

Zum Nachweis der Korrektheit des verteilten TIMEWARP-Simulators wird dieser schrittweise in einer Top-Down-Vorgehensweise entwickelt. Auf der abstraktesten Ebene wird ein einfacher, nicht verteilter Simulator spezifiziert. Diese erste formale Spezifikation ist einfach und abstrakt gehalten, da sie nicht weiter verifizierbar ist und die Basis für die gesamte Entwicklung bildet. Sie ist als tabellarische Spezifikation in Abbildung 4 angegeben. Der Datenzustand ist demnach zu jedem Zeitpunkt gegeben durch den Zustand s des Modells, der Ereignismenge ev und den aktuellen Simulationszeitpunkt vt. In jedem Schritt des Simulators werden eine Ereignismenge

spec Simulator : $\{\} \rightarrow O^{\underline{\infty}}$				
data	s	: $STATES$	=	$START$
	ev	: $EVENTS$	=	EV
	vt	: $TIME^{\infty}$	=	$nxt(EV, 0)$

O	**postcondition**
$\Pi_0(sim(s, ev, vt))$	$ev' = ev \cup sim(s, ev, vt)$ $s' = next(s, ev, vt)$ $vt' = nxt(ev', vt)$

Abbildung 4: Spezifikation des abstrakten Simulators

auf dem Kanal O ausgegeben und der Nachfolgezustand errechnet. Das die eigentliche Simulation betreffende Verhalten ist in Hilfsfunktionen verborgen, deren Eigenschaften algebraisch spezifiziert werden. So wird beispielsweise von der Funktion sim durch $sim(s, \emptyset, vt) = \emptyset$ gefordert, daß bei der Simulation des Zeitpunkts vt im Zustand s keine neuen Ereignisse entstehen, wenn keine aktuellen Ereignisse vorliegen. Werden die Hilfsfunktionen später implementiert, so müssen lediglich Eigenschaften wie diese nachgewiesen werden.

Die Verfeinerung dieser Spezifikation in einen TimeWarp-Simulator umfaßt zwei wesentliche Aspekte: Zunächst wird in zwei Schritten ein *verteilter* Simulator entwickelt, dessen Komponenten aber noch konservativ synchronisiert sind. Dann wird die einfache Kommunikation zwischen den Komponenten ersetzt durch die *nicht-konservative Kommunikation* des TimeWarp, die den Umgang mit verspäteten Ereignissen und Stornierungen umfaßt. Alle Verfeinerungen sind als Interaktionsverfeinerungen formalisiert und unter Verwendung der algebraischen Eigenschaften der Hilfsfunktionen verifiziert.

Im Laufe dieser Fallstudie wurden einige Erkenntnisse gewonnen, die hier zusammengefaßt werden: Die tabellarischen Beschreibungstechniken von Focus sind geeignet, das komplexe Verhalten des verteilten TimeWarp-Simulators sehr kompakt zu beschreiben. Mit Hilfe der semantischen Basis und inbesondere der modularen Verfeinerungsbegriffe läßt sich die Korrektheitsaussage formal nachweisen. Dies untermauert den Nutzen einer eindeutigen Semantik für Beschreibungstechniken. Die Beweise sind nicht einfach, was aufgrund der Komplexität des TimeWarp nicht überraschen kann. Allerdings muß in formalen Beweisen auch für vermeintlich offensichtliche Aussagen viel technischer Aufwand betrieben werden. Abhilfe wird durch die geplante Integration von Beweiswerkzeugen in Focus erreicht werden. Dafür konnten bei der Beweissuche Fehler in den Spezifikationen und vernachlässigte Forderungen an die Hilfsfunktionen aufgedeckt werden, die sonst vermutlich nicht aufgefallen wären. Dies bestätigt die Aussage, daß durch formale Spezifikation und Verifikation die Qualität einer Systementwicklung deutlich erhöht werden kann.

3.3 Semantische Fundierung von SDL

Eine Anwendung von Focus für die semantische Fundierung einer industriell verbreiteten Spezifikationssprache stellt [5] vor. Im Rahmen von Focus wird eine formale Semantik für SDL

(Specification and Description Language) und darauf aufbauend eine Methode für die Verifikation von SDL-Spezifikationen entwickelt, so daß SDL zukünftig als formale Spezifikationssprache im Systementwicklungsprozeß eingesetzt werden kann. SDL ist eine in der Industrie weit verbreitete und von der ITU-T standardisierte Spezifikationssprache. Sie ist auf die Modellierung von ereignisgesteuerten und interaktiven verteilten Systemen ausgerichtet, die nebenläufige Aktivitäten und asynchrone Kommunikation über Nachrichtenaustausch aufweisen. Bei SDL handelt es sich um eine informelle Spezifikationssprache, bei der den überwiegend graphischen Sprachmitteln eine nur ungenügend definierte informelle Semantik gegenüber steht. Dies hat zur Folge, daß die Bedeutung von SDL-Spezifikationen oft unklar und widersprüchlich ist, was vor allem für das Zeitkonzept von SDL gilt. Darüber hinaus ist mangels einer formalen Semantik, die auf mathematisch-logischen Konzepten basiert, die Durchführung von Verifikationsaufgaben nicht möglich.

Um die formale Semantik für die SDL-Spezifikation zu erhalten, wird diese in eine Spezifikation nach FOCUS übersetzt, wobei das in FOCUS beschriebene System das gleiche Verhalten wie das mit SDL beschriebene System aufweist. Für das FOCUS-System ist die Semantik basierend auf mathematisch-logischen Konzepten durch stromverarbeitende Funktionen gegeben. Dadurch erhält die SDL-Beschreibung eine wohldefinierte formale Semantik. Basierend auf dieser Semantik wird eine Methode für die Verifikation von SDL-Spezifikationen entwickelt, die sich auf die Beweistechniken von FOCUS stützt.

Die *Semantikdefinition* für SDL gliedert sich gemäß der Aspekte Struktur, Datenanteil und Verhalten. Der strukturelle Aufbau des gegebenen SDL-Systems wird in FOCUS durch ein Netzwerk von Komponenten beschrieben. Jedem SDL-Block und jedem SDL-Prozeß wird eine Komponente in FOCUS zugeordnet; die Hierarchieebenen sowie die Verbindungsstruktur aus dem SDL-System werden nach FOCUS übertragen und mit ANDL spezifiziert. Der Datenanteil einer SDL-Spezifikation umfaßt die Definition abstrakter Datentypen, die Deklaration von lokalen Variablen in den SDL-Prozessen und den Zugriff auf die Variablen während des Prozeßablaufs. Für die Fundierung dieses Spezifikationsanteils wird eine algebraische Spezifikationssprache verwendet.

Den umfangreichsten Teil der Semantikdefinition stellt die formale Fundierung der SDL-Prozesse dar, die das Verhalten des Systems bestimmen. Dabei genügt es nicht, allein die graphische Darstellung des erweiterten Zustandsautomaten zu betrachten, der durch einen SDL-Prozeß gegeben ist. Auch die Konzepte, die nicht explizit in einer SDL-Prozeßspezifikation dargestellt sind, tragen maßgeblich zum Verhalten des Prozesses bei. Dazu zählen der unbeschränkte Eingabepuffer des Prozesses, in den die ankommenden Signale eingereiht werden, und die Verzögerung von Timern, die der Prozeß während eines Zustandsübergangs setzen kann. Das Verhalten eines SDL-Prozesses wird deshalb nicht durch eine einzelne Komponente, sondern durch eine Menge von Komponenten modelliert, die durch ihr Zusammenwirken das Verhalten des SDL-Prozesses erbringen. Dazu zählt beispielsweise neben einer Komponente *Fair Merge*, die den Eingabepuffer des Prozesses modelliert, eine Komponente *PR*, die den Eingabestrom, den sie von *Fair Merge* erhält, gemäß des Zustandsautomaten verarbeitet. Dabei wird jedem Kontrollzustand Z eine Funktion f_Z zugeordnet. Deren Verhalten wird konstruktiv durch eine Reihe von rekursiven Funktionsgleichungen festgelegt, die aus den Zustandsübergängen des SDL-Prozesses abgeleitet werden. Das Ein-/Ausgabeverhalten des Prozesses entspricht der Funktion, die den Anfangszustand des Prozesses modelliert; diese verarbeitet das erste Signal

des Eingabestroms gemäß dem gegebenen Zustandsübergang und ruft für die Verarbeitung des restlichen Stroms die Funktion für den Folgezustand auf. Eine Funktionsgleichung für einen Zustandsübergang von Z nach Z' im SDL-Prozeß entspricht folgendem Schema:

$$f_Z\ [\sigma]\ (a\ \&\ in)\ =\ b_1\ \&\ \ldots\ \&\ b_n\ \&\ f_{Z'}\ [\sigma']\ (in)$$

Die Funktion f_Z erhält als Eingabe den Eingabestrom $a\&in$ und erzeugt abhängig vom ersten Element a des Eingabestroms eine Folge von Ausgabesignalen $b_1, \ldots, b_n$, die durch den Operator & zu einem Strom konkateniert werden. Mit dem restlichen Eingabestrom in wird die Funktion $f_{Z'}$ aufgerufen, die den Folgezustand Z' repräsentiert. Mögliche Veränderungen des Datenzustands werden durch den Übergang von σ nach σ' ausgedrückt.

Mit dieser Vorgehensweise wird einer SDL-Spezifikation eine Menge von stromverarbeitenden Funktionen als formale Semantik zugewiesen. Für SDL-Systeme mit dynamischer Prozeßgenerierung während des Systemablaufs wird für die Semantikdefinition das FOCUS-Modell für mobile, dynamische Systeme verwendet. Die Semantik ist in diesem Fall eine Menge von mobilen Funktionen. An der prinzipiellen Vorgehensweise ändert sich dabei nichts; die dynamische Prozeßerzeugung während eines Zustandsübergangs wird durch einen rekursiven Funktionsaufruf in den Funktionsgleichungen modelliert.

Die formale Semantik ist Ausgangspunkt für die *Verifikation* von Eigenschaften einer SDL-Spezifikation. FOCUS bietet aufgrund seiner mathematisch-logischen Fundierung Beweisprinzipien der funktionalen Logik und der Bereichstheorie. Für die Eigenschaftsverifikation von SDL-Spezifikation ergibt sich folgender durchgängiger Ansatz:

1. Entwicklung der SDL-Spezifikation.
2. Angabe der formalen Semantik der SDL-Spezifikation als FOCUS-Systemspezifikation S gemäß dem oben beschriebenen Vorgehen.
3. Formulierung der Beweisverpflichtung: Eigenschaften, die das mit SDL spezifizierte System S erfüllen soll, werden in FOCUS als Prädikat über der SDL-Semantik formuliert. Es wird gefordert, daß jede Funktion f_S, die ein zulässiges Ein-/Ausgabeverhalten (i, o) der SDL-Spezifikation festlegt, die im Prädikat P angegebenen Eigenschaften erfüllt. Seien I die Menge der Eingabenachrichten, O die Menge der Ausgabenachrichten von S.

$$\forall f_S \in [\![S]\!],\ i \in I^{\underline{\infty}},\ o \in O^{\underline{\infty}} : f_S(i) = o\ \Rightarrow\ P(i, o)$$

4. Durchführung der Verifikationsaufgabe: Die vorgesehene Anbindung von FOCUS an interaktive Theorembeweiser wird zukünftig eine maschinengestützte Beweisführung ermöglichen.

Die Semantikdefinition von SDL zeigt, daß FOCUS geeignet ist, um die Konzepte industrieller Beschreibungstechniken zu formalisieren. Für die Systementwicklung in FOCUS steht nun mit SDL eine Zielsprache zur Verfügung, die in der Praxis auf hohe Akzeptanz stößt.

4 Zusammenfassung und Ausblick

Mit der vorliegenden Arbeit wurden die semantischen Grundlagen, die Beschreibungstechniken sowie das Verfeinerungskonzept der formalen Methodik FOCUS skizziert. Die Beschreibung

aktueller Anwendungen aus unterschiedlichen Themenbereichen zeigt die Praktikabilität und drei Einsatzmöglichkeiten für FOCUS: die systematische *Spezifikationserstellung* mit formalen Techniken, die Durchführung von als korrekt bewiesenen *Verfeinerungsschritten* zur Konkretisierung des Systemverhaltens sowie die *semantische Fundierung* der Sprache SDL einschließlich einer methodischen Anleitung zum Führen von Eigenschaftsbeweisen zu SDL-Spezifikationen. Die hier vorliegende FOCUSBeschreibung soll dazu beitragen, das Interesse an der Methodik zu wecken und es in einem größeren Umfeld bekannt zu machen. Zukünftige Arbeiten zielen darauf ab, FOCUS auf dessen praktischen Einsatz vorzubereiten und Anwendern verstärkt methodische Unterstützung anzubieten.

Der Umgang mit formalen Methoden erfordert vor allem für die Durchführung von Verifikationsaufgaben ein hohes Maß an mathematisch-logischen Grundkenntnissen. Um diese Aufgabe zu erleichtern, muß der Einsatz graphischer Beschreibungstechniken ermöglicht werden, so daß die semantisch-orientierten Formalismen dem Entwickler von Spezifikationen weitgehend verborgen bleiben. Ferner müssen eine Anbindung von automatischen oder interaktiven Beweiswerkzeugen erfolgen sowie methodische Anleitungen für Systementwicklungen gegeben werden. Die Weiterentwicklung des prototypischen Werkzeugs AUTOFOCUS ist in diesem Sinne voranzutreiben. Aufgrund der Vielzahl an Einsatzmöglichkeiten von FOCUS mit seinen unterschiedlichen Vorgehensweisen und Beschreibungstechniken ist es für den Anwender schwierig, eine für seine Problemstellung geeignete Vorgehensweise zu finden. Daher ist es unerläßlich, allgemeine Richtlinien und ein FOCUS-Vorgehensmodell festzulegen sowie eine ingenieurtechnisch ausgerichtete Beschreibung für den Einsatz von FOCUS zu erstellen. Erfahrungen hierzu können vor allem durch die Verwendung von FOCUS in weiteren Anwendungsbereichen und dem dabei erhaltenen Feedback gesammelt werden.

Danksagung

Wir bedanken uns bei Peter Paul Spies und Andreas Wolf für ihre konstruktive Kritik zu einer Vorversion dieses Papiers.

Literatur

[1] Homepage des FOCUS-Projektes mit allen relevanten Veröffentlichungen. Verfügbar im World Wide Web unter http://www4.informatik.tu-muenchen.de/proj/focus/.

[2] M. Broy, M. Breitling, B. Schätz, und K. Spies. Summary of Case Studies in FOCUS - Part II. Technischer Bericht TUM-I9740, Technische Universität München, 1997.

[3] K. Spies. Eine Methode zur formalen Modellierung von Betriebssystemkonzepten. Technischer Bericht TUM-I9809, Technische Universität München, 1998. Dissertation.

[4] M. Breitling. Formalizing and Verifying TIMEWARP with FOCUS. Technischer Bericht TUM-I9744, Technische Universität München, 1997.

[5] U. Hinkel. Formale, semantische Fundierung und eine darauf abgestützte Verifikationsmethode für SDL, 1998. Dissertation, in Vorbereitung.