Entwurf verteilter Systeme im Sonderforschungsbereich 342

Jan Philipps *
philipps@in.tum.de

Institut für Informatik
Technische Universität München
80290 München

1 Einführung

Seit seiner Gründung im Jahr 1990 werden im Sonderforschungsbereich 342 „Werkzeuge und Methoden für die Nutzung paralleler Rechensysteme“ Techniken zur Unterstützung der verschiedenen Entwurfsphasen wie auch der Ausführung paralleler Systeme entwickelt. Ziel ist es, die Programmierung verteilter und paralleler Systeme beherrschbar und ihre Nutzung effizient und zuverlässig zu machen.

Die Arbeiten erfolgen unter Berücksichtigung der ganzen Bandbreite von Rechnerarchitekturen, Programmiermodellen und Kommunikations- und Parallelisierungskonzepten. Der Schwerpunkt des SFB 342 liegt auf Systemen mit verteilten Speichern, da dieses Modell einerseits mit dem zunehmenden Einsatz von Workstation-Netzen an Bedeutung gewinnt, andererseits durch das Programmiermodell der nachrichtengekoppelten Kommunikation am weitesten von der klassischen sequentiellen Programmierung entfernt ist.

Die im SFB entwickelten Techniken werden in unterschiedlichsten Anwendungsbereichen erprobt, teils an der TU München selbst, teils in industriellen Begleitprojekten sowie in Kooperation mit externen industriellen Partnern.

Die Arbeiten des SFB haben auch Auswirkungen über die unmittelbaren Projektpartner hinaus: So gibt es Kooperationen mit Graduiertenkollegs, dem Leibniz-Rechenzentrum und dem von der Bayerischen Forschungsstiftung geförderten Verbundprojekt FORSOFT.

Der nächste Abschnitt gibt einen Überblick über die Struktur des SFB 342. Der Schwerpunkt dieses Berichts ist die Darstellung der Arbeiten des Teilprojekts A6, das sich vornehmlich mit mathematischen fundierten Entwurfstechniken für verteilte Systeme beschäftigt: In §3 werden die in A6 entwickelten Techniken kurz vorgestellt, §4 beschreibt mit der Architekturverfeinerung von verteilten Systemen eine Anwendung dieser Techniken.

Weitere Informationen finden sich auf den WWW-Seiten des SFB 342:
`http://www.informatik.tu-muenchen.de/~sfb342`

* Diese Arbeit entstand im Teilprojekt A6 des SFB 342 der Deutschen Forschungsgemeinschaft.

Sonderforschungsbereich 342		
A Grundlagen und Werkzeuge	**B** Methoden und Anwendungen	**C** Industrielle Partner
A1 Werkzeugumgebungen für parallele und verteilte Systeme **A3** Spezifikation, Analyse und Modellierung **A5** Parallelisierung in Inferenzsystemen **A6** Entwurfsmethodik für verteilte Systeme **A7** Effiziente parallele Algorithmen und Schedules **A8** Konstruktion heteromorph paralleler Systeme	**B1** Parallelisierung von Entwurfsverfahren für höchstintegrierte Schaltungen **B2** Parallelisierung von Datenbanksystemen **B3** Parallelisierung hierarchisch strukturierter numerischer Algorithmen **B4** Parallele Simulation digitaler Systeme mit hoher Komplexität	**C1** Lastverteilung und Fehlertoleranz verteilter Systeme **C3** Parallele und verteilte Simulation komplexer Systeme

Abbildung 1. Struktur des SFB 342

2 Struktur des SFB 342

Der SFB 342 gliedert sich in drei Teilbereiche (Abbildung 1). Projektpartner sind Lehrstühle der Fakultät für Informatik der TU München, der Lehrstuhl für rechnergestütztes Entwerfen der Fakultät für Elektrotechnik und die Zentralabteilung Forschung und Technik der Siemens AG.

Projektbereich A: Grundlagen und Werkzeuge für die Virtualisierung paralleler Architekturen

In diesem Bereich werden Methoden zur Parallelisierung und Verteilung und Techniken für den Entwurf und die Analyse paralleler und verteilter Verfahren und Architekturen entwickelt.

Die Teilprojekte des Bereichs A decken dabei alle Entwurfsebenen ab, von Modellierungs- und Spezifikationstechniken für verteilte Anwendungen über Analyse- und Verifikationstechniken für Komponenten verteilter Systeme bis zur Entwicklung von Schedulingstrategien für unterschiedliche Netzwerktopologien. Für die spezielleren Anforderungen der Parallelisierung von Theorembeweisern und Schedulingalgorithmen werden eigene Verteilungsstrategien entwickelt.

Zur Unterstützung der späteren Entwurfsebenen und der Ausführung paralleler Systeme werden Werkzeuge – etwa zum Monitoring und zur Leistungsbewertung – entwickelt, die sich für nachrichtengekoppelte Systeme wie auch für Systeme mit verteiltem gemeinsamen Speicher, etwa SCI-basierten PC-Clustern, einsetzen lassen.

Projektbereich B: Methoden und Anwendungen für die Virtualisierung paralleler Architekturen

In diesem Projektbereich werden die im Bereich A entwickelten Grundlagen und Methoden anhand praxisnaher Anwendungen evaluiert und prototypisch genutzt. Umgekehrt beeinflussen die dabei gewonnenen Erkenntnisse die Arbeiten im Projektbereich A.

Ziel der Anwendungsprojekte ist es einerseits Konzepte zu evaluieren, andererseits aber auch konkrete parallele Implementierungen für erstellen, die für praktische Probleme bessere Lösungen schneller berechnen, als dies mit herkömmlichen sequentiellen Systemen möglich ist. Die Projekte in diesem Bereich behandeln daher mit Schaltungsentwurf, Simulation, Datenbanken und numerischen Algorithmen Gebiete, bei denen herkömmliche Techniken schnell an ihre Grenzen stoßen.

Projektbereich C: Industrielles Begleitprojekt

Die Projekte dieses Bereichs werden von der Zentralabteilung Forschung und Technik der Siemens AG durchgeführt.

Im Bereich C werden Fragen zur Lastverteilung in heterogenen verteilten Systemen behandelt. Ein zentrales Thema ist der Einsatz von Middleware wie CORBA. Der seit einiger Zeit zunehmende Einsatz mobiler Systeme erfordert dabei neue Paradigmen und Techniken, um Lastverteilung und Fehlertoleranz auch bei temporären Verbindungen zu erreichen.

Die Siemens AG untersucht zudem Partitionierungstechniken zur Simulation hochkomplexer Systeme. Ein konkretes Problem ist etwa die Simulation von Schaltkreisen auf Transistorebene, die mit sequentiellen Systemen nur unbefriedigend langsam erfolgt.

Querschnittsthemen

Orthogonal zu der dargestellten Projektstruktur werden in einer Reihe von Querschnittsthemen Fragen von übergreifendem Interesse behandelt. Die Teilprojekte kooperieren darin in Themen wie Lastverteilung, Entwicklung heuristischer Techniken und Beweisunterstützung.

3 Teilprojekt A6

Im Teilprojekt A6 wird mit Focus eine Methodik zur mathematisch fundierten Entwicklung verteilter Systeme entwickelt [5, 2]. Im Sinne von Focus besteht ein System aus einem Netz von interagierenden Komponenten, die über Nachrichtenkanäle kommunizieren. Das Verhalten einer Systemkomponente wird durch die Relation zwischen der Folge der ein- und ausgehenden Nachrichtenfolge festgelegt.

Mathematische 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.

Das Verhalten einer Komponente ist eine Relation zwischen Eingangs- und Ausgangsströmen; es wird durch ein Prädikat spezifiziert, dessen freie Variablen mit Ein- und Ausgangsströmen belegt werden. Dabei gibt es Randbedingungen, die sicherstellen daß die Komponentenrelation total ist (für jede mögliche Eingangsnachrichtenfolge gibt es Ausgabefolgen) und daß sie kausal korrekt ist (Ausgaben einer Komponente dürfen nicht abhängig sein von Eingaben zu einem späteren Zeitpunkt).

Unter diesen Randbedingungen lassen sich Komponenten zu Systemen zusammenfügen. Dazu stehen parallele und sequentielle Komposition sowie Rückkopplung zur Verfügung. Das Ein-/Ausgabeverhalten eines zusammengesetzten Systems ergibt sich eindeutig aus dem Verhalten der einzelnen Komponenten durch Konjunktion der Komponentenprädikate.

Beschreibungstechniken. Für die eigentliche Modellierung von Systemen bietet Focus eine Reihe von Beschreibungstechniken, die es ermöglichen, ein System aus verschiedenen Sichten zu beschreiben. Jede dieser Beschreibungstechniken hat eine mathematisch-logisch definierte präzise Semantik. Mit den im folgenden kurz skizzierten Beschreibungstechniken kann damit spezifiziert werden, ohne die mathematischen Formalismen explizit verwenden zu müssen.

Mit *Systemstrukturdiagrammen* (SSDs) wird die Kommunikationsstruktur eines Systems durch Datenflußdiagramme, gegeben durch die Kanalverbindung der Komponenten sowie ihre syntaktischen Schnittstelle, definiert. Neben einer graphischen Darstellung sind auch textuell orientierten Darstellungsformen durch Operatoren und Gleichungen möglich. *Zustandsübergangsdiagramme* (STDs) beschreiben das Verhalten von Komponenten als erweiterter endlicher Automat. Für die Übergänge werden jeweils Vor- und Nachbedingungen auf den (Daten-)Zustandsvariablen sowie Ein- und Ausgabeaktionen auf den Kanälen definiert. Durch *Erweiterte Ereignisdiagramme (EETs)* werden Abläufe des Gesamtsystems als Interaktionen zwischen den Systemkomponenten angegeben.

Die genannten Beschreibungstechniken werden von dem CASE-Tool AutoFocus unterstützt [6, 1]; mit AutoFocus können Systeme modelliert, animiert und über eine Multimediaschnittstelle visualisiert werden. AutoFocus erlaubt auch Konsistenzprüfungen zwischen verschiedenen Dokumenten und hat erste prototypische Schnittstellen zu Verifikationswerkzeugen und zur Codegenerierung.

Verfeinerungen mit Focus. Bei einer formalen Entwicklung komplexer, verteilter Systeme wird der gewünschte Detaillierungsgrad ausgehend von einer abstrakten Spezifikation durch eine schrittweise Verfeinerung des Systems bis hin zu einer Implementierung erreicht. Focus bietet hierfür die folgenden Verfeinerungskonzepte an [3, 4]:

Verhaltensverfeinerung reduziert Unterspezifikation. Was auf abstrakter Ebene noch irrelevant und daher unspezifiziert war, wird dabei durch Entwurfsentscheidungen konkretisiert.

Strukturelle Verfeinerung verfeinert den statischen Aufbau des verteilten Systems, indem Komponenten durch Netzwerke von interagierenden Komponenten ersetzt werden.

Schnittstellenverfeinerung verändert Anzahl und Typen der Kommunikationskanäle. Eine typische Anwendung für Schnittstellenverfeinerung ist das Ersetzen von Datentypen für die Kommunikation zwischen Komponenten durch implementierungsnähere Typen.

Für alle Varianten stehen Beweisregeln zur Verfügung, die die Vorbedingungen für die Verfeinerungsschritte festlegen.

4 Architekturverfeinerung

Die Techniken des Teilprojekts A6 dienen dem Entwurf verteilter Systeme durch schrittweise Verfeinerung von abstrakten Spezifikationen hin zu konkreten Implementierungen. Bei jedem Verfeinerungsschritt werden neue Entwurfsentscheidungen getroffen.

In der Praxis ist es zusätzlich zu dem Entwurf neuer System oft erforderlich, aufgrund neuer Anforderungen ein existierendes System zu modifizieren, oder nachträglich Entwurfsentscheidungen zu revidieren, etwa um die Effizienz zu steigern. Die oben erwähnten Verfeinerungskonzepte und -regeln eignen sich für nachträgliche Veränderungen an bereits bestehenden Systemen und Systemarchitekturen weniger.

Solche *evolutionären* Systemveränderungen erfordern zunächst einen präzisen Architekturbegriff für Softwaresysteme. Für Datenflußarchitekturen [10] bietet Focus die nötigen Grundlagen. Erlaubte Veränderungen an einem System können dann mit eigenen Architekturverfeinerungsregeln beschrieben werden, die ein *Glasbox*-Verfeinerungskonzept darstellen: Komponentennetzwerke können durch Komponentennetzwerke ersetzt werden.

Eine genauere Darstellung der verwendeten Techniken und ihrer mathematischen Grundlagen findet sich in [7, 8].

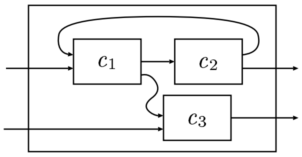

Abbildung 2. Datenflußkomponente und -system

Komponenten und Systeme. Eine Datenflußsystem (Abbildung 2) besteht aus Eingabe- und Ausgabekanälen sowie einer Menge von Komponenten und einer Verbindungsstruktur, die folgenden Anforderungen genügen muß: Komponenten haben keine gemeinsamen Ausgabekanäle, jeder Komponenteneingangskanal ist entweder Ausgabe einer anderen Komponente oder Eingabe des Systems, kein Systemeingangskanal ist Ausgabekanal einer Komponente und jeder Ausgabekanal des Systems ist auch Ausgabekanal einer Komponente.

Unter diesen Umständen kann eine Blackbox-Sicht des Systems gebildet werden, indem nur das Verhalten auf den Systemeingängen und -ausgängen berücksichtigt wird, nicht aber die interne Struktur.

Verfeinerungsregeln. Die in §3 erwähnten Verfeinerungsregeln basieren auf Systembeschreibungen über eine abstrakte Syntax aus Komponentenspezifikationen und Kompositionsoperatoren. Im Gegensatz dazu nehmen die Architekturregeln Bezug auf einzelne Kanäle und Komponenten: Oft müssen bestehende Systeme verändert werden, für die die abstrakte Strukturbeschreibung nicht mehr vorliegt.

Die Architekturregeln erlauben folgende Veränderungen eine Systems:

- Einführen und Entfernen von Komponenten aus einem System,
- Einführen und Entfernen von Eingabekanälen einer Komponente,
- Einführen und Entfernen von Ausgabekanälen einer Komponente,
- Verfeinerung des Komponentenverhalten, ggf. unter Einbeziehung einer Invariante über das Systemverhalten,
- Ersetzen einer Komponente durch ein Komponentennetzwerk und umgekehrt.

Die Regeln erhalten die oben erwähnten Anforderungen an die Systemarchitektur. Jede dieser Regeln hat Vorbedingungen, die teils syntaktischer Natur sind, teils Aussagen über das Verhalten der betroffenen Komponente oder der Komponenten in ihrer Umgebung erfordert. Die Regeln, ihre Vorbedingungen und ihre mathematische Rechtfertigung auf der Grundlage von Focus finden sich in [8].

Beispiel. Abbildung 3 zeigt die Architekturverfeinerung eines einfachen Datenerfassungssystems. Die Komponente PRE sammelt Daten, führt einige Berechnungen auf ihnen durch und schickt sie zu einer entfernten Datenbank

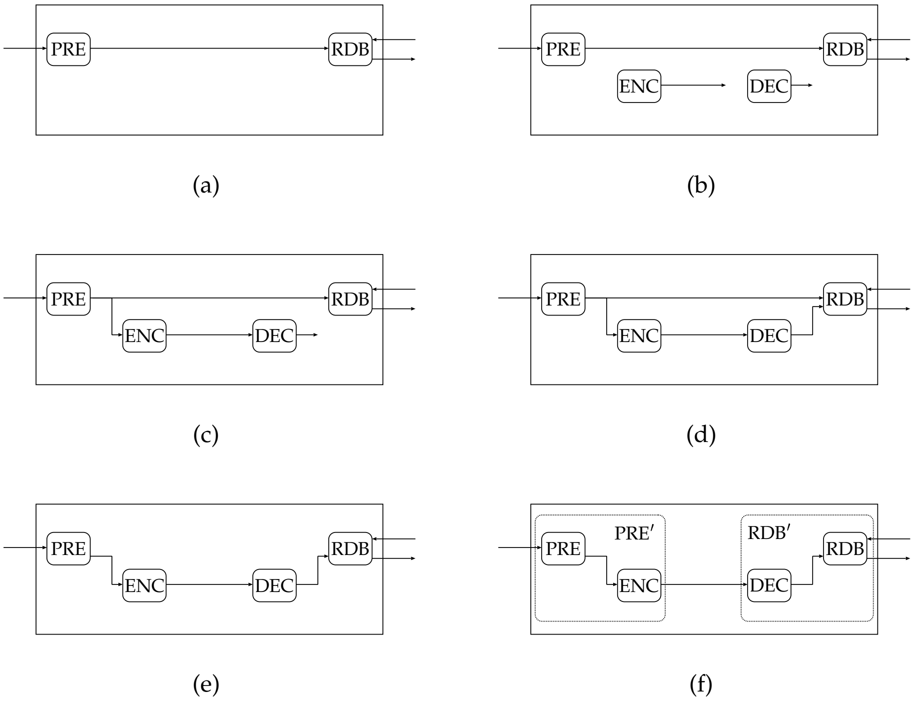

Abbildung 3. Architekturverfeinerung

RDB. Um die Übertragungszeit der Daten zu verringern, sollen in einer neuen Version des Systems nur die Differenz der aktuellen Daten zu den vorher gesendeten Daten übertragen werden. Die sechs Strukturdiagramme zeigen die Verfeinerungsschritte, die dazu erforderlich sind: Einführen der Codierungs- und Decodierungskomponenten ENC und DEC, Verbinden dieser Komponenten mit dem bestehenden System, Entfernen der bisherigen Verbindung zwischen PRE und RDB und Kombinieren von PRE und ENC sowie RDB und DEC zu neuen Erfassungs- und Speicherkomponenten.

Die meisten dieser Schritte sind rein syntaktisch; nur der Schritt von (d) zu (e), bei dem die bestehende Verbindung gelöst wird, erfordert eine formale Rechtfertigung: Es muß gezeigt werden, daß die codierten Daten im wesentlichen dieselbe Information enthalten wie die ursprüngliche Verbindung. Der Beweis dazu findet sich in [8].

Das Beispiel zeigt auch, wie sich die Architekturverfeinerungsregeln zu komplexeren anwendungsorientierten Verfeinerungsmustern zusammensetzen lassen. Es werden nur sehr schwache Annahmen über das Verhalten von PRE und RDB gemacht, so daß sich dieses Beispiel ohne weitere Beweisverpflichtung auf ähnliche Situationen anwenden läßt.

5 Zusammenfassung und Ausblick

Das im Teilprojekt A6 entwickelte Focus umfaßt mathematische Grundlagen, Beschreibungstechniken sowie Verfeinerungs- und Evolutionstechniken für verteilte Systeme. Mit diesen Techniken lassen sich einerseits Systeme ausgehend von ersten abstrakten Anforderungen entwerfen, andererseits aber auch bestehende Systemarchitekturen verändern, um sie an neu enstandene Anforderungen anzupassen.

Ein Schwerpunkt bei den aktuellen Arbeiten zu Focus ist die Anpassung der Techniken an die immer mehr an Bedeutung gewinnenden *eingebetteten Systeme*. Bei diesen Systemen steht der Softwareanteil nicht im Vordergrund: Er erfüllt eingebettet in elektrische oder mechanische Systeme Steuerungsaufgaben. Eingebettete Systeme müssen besondere Randbedingungen erfüllen, sowohl für die einzelnen Produkte (Speicher- und Rechenzeitbeschränkungen), wie auch für den Entwicklungsprozeß (Verwendung von Prototypen, Entwurfsentscheidungen im HW/SW-Codesign). Insbesondere zielen die aktuellen Arbeiten auf spezielle Verfeinerungsmuster für eingebettete Systeme, auf die automatische Verifikation der Systeme mit Modellprüfern und auf Codegenerierungstechniken [9].

Danksagung. Die Arbeiten zur Architekturverfeinerung erfolgen gemeinsam mit Bernhard Rumpe. Ich danke Katharina Spies und Thomas Ludwig für die konstruktive Kritik zu einer Vorversion dieses Berichts.

Literatur

[1] Das AutoFocus-Projekt. http://autofocus.informatik.tu-muenchen.de/.

[2] M. Breitling, U. Hinkel, and K. Spies. Formale Entwicklung verteilter reaktiver Systeme mit Focus. In *Formale Beschreibungstechniken für verteilte Systeme, 8. GI/ITG Fachgespräch*, 1998.

[3] M. Broy. Compositional refinement of interactive systems. Working Material, International Summer School on Program Design Calculi, August 1992.

[4] M. Broy. (Inter-) action refinement: the easy way. Working Material, International Summer School on Program Design Calculi, August 1992.

[5] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber. The Design of Distributed Systems: An Introduction to Focus—Revised Version. Technical Report TUM-I9202-2, Institut für Informatik, Technische Universität München, 1993.

[6] F. Huber, B. Schätz, A. Schmidt, and K. Spies. Autofocus—a tool for distributed systems specification. In *Proceedings FTRTFT'96 — Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 1135, 1996.

[7] J. Philipps and B. Rumpe. Refinement of information flow architectures. In *ICFEM'97*, 1997.

[8] J. Philipps and B. Rumpe. Refinement of pipe and filter architectures. In *FM'99*, 1999. To appear.

[9] J. Philipps and A. Schmidt. Enwurf und Implementierung eingebetteter Systeme. In *Formale Beschreibungstechniken für verteilte Systeme, 8. GI/ITG Fachgespräch*, 1998.

[10] M. Shaw and D. Garlan. *Software Architecture*. Prentice Hall, 1996.