Jahresbericht 2012 des Lehrstuhls für Informatik 2 (Programmiersysteme)

Anschrift: Martensstr. 3, 91058 Erlangen
Tel.: +49-9131-85-27621
Fax: +49-9131-85-28809
E-Mail: info@i2.informatik.uni-erlangen.de

Ordinarius:
Prof. Dr. Michael Philippsen
Honorarprofessor:
Hon.-Prof. Dr.-Ing. Bernd Hindel
Hon.-Prof. Dr.-Ing. Detlef Kips
Emeritus:
Prof. em. Dr. Hans Jürgen Schneider
Sekretariat:
Agnes Brütting
Wiss. Mitarbeiter:
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Daniel Brinkers
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Stefan Kempf
Dipl.-Math. Jakob Krainz
Andreas Kumlehn, M. Sc.
Dipl.-Inf. Christopher Mutschler
Dr.-Ing. Norbert Oster
Norbert Tausch, M. Eng.
PD Dr. Ronald Veldema
Dipl.-Inf. Tobias Werth
Gast:
Dr.-Ing. Josef Adersberger
Dipl.-Inf. Samir Al-Hilank
Sören Braunstein, M. Sc.
Dipl.-Inf. Ralf Ellner
Dr.-Ing. Martin Jung
Externes Lehrpersonal:
Dr.-Ing. Klaudia Dussa-Zieger
Dr.-Ing. Stephan Otto

Der 1972 gegründete Lehrstuhl Informatik 2 (Programmiersysteme) wird seit April 2002 von Prof. Dr. Michael Philippsen (als Nachfolger von Prof. Dr. em. H.-J. Schneider) geleitet. Eng mit dem Lehrstuhl assoziiert sind die beiden Professuren für Didaktik der Informatik und Open Source Software, deren Forschungsarbeiten separat dargestellt sind.

1 Forschungsschwerpunkte

Im Mittelpunkt der **Programmiersystemforschung** des Lehrstuhls stehen parallele und verteilte Systeme und deren Programmierung sowie Programmiersysteme für eingebettete und mobile Systeme. Software (und deren Erstellung) für solche Systeme sollte nicht komplexer, aber genauso portabel, wartbar und robust sein, wie heute schon für Einprozessorsysteme und Arbeitsplatzrechner. Langfristiges Ziel ist es, den Anwendungen die verfügbare Rechen- und Kommunikationsleistung möglichst ungebremst zur Verfügung zu stellen bzw. aus begrenzten Systemen ein Maximum herauszuholen. Ein besonderer Arbeitsschwerpunkt sind Programmiersysteme für Multicore-Rechner, da deren unausweichliche Verbreitung ebenso wie die preisgünstige Verfügbarkeit von sehr leistungsstarker paralleler Spezialhardware im Massenmarkt (z.B. Grafik-Karten oder FPA-Hardware) kaum abschätzbare Auswirkungen auf die Software-Landschaft haben werden. Forschungsergebnisse werden stets an eigenen Prototypen und Demonstratoren praktisch evaluiert.

Wichtige Forschungsfelder

- **Vorhandenes Parallelisierungspotential ausschöpfen.** Da die Taktraten von Mehrkernrechnern kaum noch steigen, dafür aber deren Kernanzahl anwachsen wird, muss das Parallelisierungspotential existierender Software erkannt und ausgeschöpft werden, um an den Leistungssteigerungen der Hardware teilzuhaben. Außer vielleicht in Nischen führt kein Weg an einem Umstieg auf Parallelverarbeitung vorbei. Deshalb entwickelt der Lehrstuhl Werkzeuge, die dem Programmierer interaktiv beim Re-Engineering bestehender Anwendungen helfen, und erarbeitet Architekturmuster für neu zu entwickelnde Software-Projekte, die eine Skalierbarkeit für und damit eine Toleranz gegen wachsende Kernanzahlen aufweisen.

- **Hochleistungsanwendungen portabel machen.** Anwendungsprogrammierer erreichen nur dann bestmögliche Laufzeiten, wenn sie die Überwindung der Latenzzeiten und die explizite Kommunikation zwischen unterschiedlichen Komponenten der Systemarchitektur manuell angehen, ihren Code mit Hardware-nahen ”Tricks” spezifisch für eine Architektur optimieren und ihre Applikation per Hand

in mehrere Teile zerlegen, von denen manche z.B. auf die Grafikkarte ausgelagert werden. Der Lehrstuhl erforscht, wie durch Anhebung des Abstraktionsniveaus der Programmierung die Produktivität gesteigert und die Portabilität verbessert werden kann, indem Code so übersetzt wird, dass verschiedene Teile auf heterogenen Systemkomponenten nebenläufig ausgeführt werden und dass die Datenkommunikation dazwischen transparent für den Entwickler erfolgt. Eine wichtige Frage dabei ist es, wie der Programmierer sein Wissen über bestehende Lokalitätsbeziehungen programmiersprachlich ausdrücken kann, damit es effizienzsteigernd nutzbar bzw. damit Lokalität schon im Design sichtbar ist. Auf dem Weg zu diesem Ziel werden im Rahmen von Re-Engineering-Projekten Details der Hardware-Architektur vor dem Anwendungsentwickler verborgen, z.B. hinter Bibliotheksschnittstellen oder in domänenspezifischen Programmierspracherweiterungen.

- **Parallelitätsgrad dynamisieren.** Hochleistungsanwendungen werden oft für eine bestimmte Prozessorzahl entwickelt. Da die benötigten Rechnerbündelknoten vom Batch-System statisch für eine feste Zeitspanne zugeteilt werden, entstehen zwangsläufig unwirtschaftliche Reservierungslöcher. Analoge Probleme treten bei vielfädigen Anwendungen auf Multicore-Rechnern auf. Der Lehrstuhl arbeitet daher an der dynamischen Anpassung des Parallelitätsgrads durch Code-Transformationen, die die Kosten der resultierenden Datenumverteilungen berücksichtigen, sowie durch Interaktion und Zusammenarbeit mit dem Betriebssystem. Die in Programmen vorgefundenen expliziten, kontrollflussorientierten Synchronisationsmaßnahmen behindern die erforderlichen Analysen, weshalb der Lehrstuhl neue und bessere Programmkonstrukte erforscht, die die bisherigen adäquat ersetzen können und die Synchronisationserfordernisse deklarativ und an den Daten orientiert ausdrücken.

- **Parallelität testbar machen.** Im Software-Engineering nimmt Testen von jeher eine wichtige Stellung ein. Stichworte wie Testüberdeckung, Testdatengenerierung, Zuverlässigkeitsbewertung etc. gehören zum Handwerk. Leider berücksichtigt die Forschung in diesem Bereich den durch Nebenläufigkeit entstehenden Indeterminismus nur unzureichend. Der Lehrstuhl arbeitet daher an Werkzeugen zur Testdatengenerierung, die bei den zugrundeliegenden Überdeckungskriterien auch Verschränkungen nebenläufiger Programmäste berücksichtigen. Dies schließt Arbeiten an Betriebssystemschnittstellen und am Ablaufplaner ebenfalls ein. Weil ferner durch die Nebenläufigkeit der Suchraum in erheblichem Maße wächst, müssen Infrastrukturen erarbeitet werden, die es ermöglichen, Massentests auf großen Rechnerbündeln auszuführen.

- **Software-Entwicklungsprozesse verbessern.** Die heute in der Industrie praktizierte Entwicklung von komplexer, geschäfts- oder sicherheitskritischer Soft-

ware in global verteilten Teams verlangt nach der Einhaltung von wohldefinierten Software-Entwicklungsprozessen mit entsprechender Werkzeugunterstützung. Im Bereich der **praktischen Softwaretechnik** arbeitet der Lehrstuhl zusammen mit den **Honorar-Professoren Dr. Bernd Hindel und Dr. Detlef Kips**, die als Geschäftsführer zweier mittelständischer Software-Beratungsunternehmen über langjährige Praxiserfahrung in industriellen Software-Entwicklungsprojekten verfügen, daher an einer maschinen-ausführbaren Notation für die Modellierung von Software-Entwicklungsprozessen, wobei wegen der zum Einsatz kommenden vielfältigen Werkzeuge und Notationen, sowohl die (teil-) automatisierte Rückgewinnung von Nachverfolgbarkeitsinformation aus den Artefakten eines Entwicklungsprojekts, als auch die modellbasierte Entwicklung, Integration und Konfiguration von Softwarekomponenten betrachtet werden, wie sie insbesondere beim Entwurf eingebetteter Systeme für den Automobilbau üblich sind.

2 Forschungsprojekte

2.1 ErLaDeF - Embedded Realtime Language Development Framework

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
PD Dr. Ronald Veldema
Andreas Kumlehn, M. Sc.
Sören Braunstein, M. Sc.
Dipl.-Inf. Thorsten Blaß
Dipl.-Math. Jakob Krainz
Dipl.-Inf. Daniel Brinkers
Beginn: 1.1.2012
Kontakt:
PD Dr. Ronald Veldema
Tel.: +49-9131-85-27622
Fax: +49-9131-85-28809
E-Mail: ronald.veldema@fau.de

ErLaDeF ist unsere Testumgebung für die Erforschung neuer Programmiersprachen und Compiler-Techniken. Unser Ziel ist eine Infrastruktur um die Programmierung von eingebetteten parallelen Systemen, insbesondere für Echtzeitsysteme, zu vereinfachen.

Im Bereich der eingebetteten Systeme und Echtzeit-Anwendungen gibt es strikte Grenzen für den Verbrauch an Betriebsmitteln wie Hauptspeicher oder Rechenzeit. Ein typischer Steuerrechner z.B. darf für eine Regelungsaufgabe im Allgemeinen nicht beliebig viel Zeit verbrauchen, um eine Fehlfunktion des gesteuerten Systems zu verhindern. Die Hardware-Entwicklung der letzten Jahre hat dazu geführt, dass vermehrt Mehrkern-Prozessoren auch in eingebetteten Systemen eingesetzt werden. Um die damit verbundene Leistungssteigerung auszunutzen, ist es daher nötig, die Steuerungs- und Systemsoftware, die auf diesen eingebetteten Systemen laufen soll, ebenfalls zu parallelisieren. Dabei darf die Anforderungen an Echtzeitfähigkeit und Ressourcenverbrauch nicht verletzt werden.

Wir erforschen verschiedene Strategien, um die Parallelisierung zu vereinfachen. Dies schließt folgende Punkte ein: Einfache Sprachen, automatisierte Parallelisierung, Bibliotheken basierend auf Entwurfsmustern für die parallele Programmierung, umfassende und detaillierte Übersetzer-Analysen sowie Modellprüfung und Beschleunigung von Übersetzeranalysen.

- Die Reduktion der Programmiersprache auf einfache Sprachelemente (im Vergleich zu z.B. Java oder C#) ermöglicht Übersetzeranalysen die potentielle Parallelität von Anwendungen auszunutzen oder Verletzungen von Echtzeitbedingungen zu finden. Dies ist die Grundlage für weitere Techniken, die wir in diesem Projekt verfolgen. Polymorphismus und Vererbung erschweren Analysen. Seit Projektbeginn suchen wir nach Alternativen, die diese Paradigmen beibehalten ohne die Genauigkeit der Analyse zu verringern. Wir haben außerdem unterschiedliche Synchronisationsmechanismen untersucht, wie z.B. transaktionalen Speicher, implizite Synchronisation und entfernte Methodenaufrufe.

- Aktuell konzentrieren wir uns bei der automatisierten Parallelisierung auf Laufzeit-Parallelisierung. Das zu parallelisierende Programm wird während seiner Ausführung auf Schleifen untersucht, die parallel ausführbar sind. Unser Ansatz besteht darin, laufzeitintensive Schleifen zunächst in zwei Durchläufen auf Parallelisierbarkeit zu untersuchen. Die Analyse-Durchläufe werden parallel ausgeführt. Im ersten Analysedurchlauf werden alle Schreibzugriffe auf den Hauptspeicher in einer gemeinsam genutzten Datenstruktur die Adresse gespeichert. Zugriffe auf diese Datenstruktur müssen nicht synchronisiert werden, wenn sichergestellt wird, dass bei konkurrierenden Schreibzugriffen ein Wert geschrieben wird. Im zweiten Durchlauf wird für jeden Speicherzugriff (auch lesende!) überprüft, ob eine Datenabhängigkeit zu einem Schreibzugriff besteht. Falls keine Datenabhängigkeiten gefunden werden, wird die Schleife parallel ausgeführt - anderenfalls sequentiell. Wir haben mit Hilfe dieser Methode bereits Leistungssteigerungen erzielt. Im vergangenem Jahr haben wir das verwendete Analyseframework erweitert. Wir haben u.A. dafür gesorgt, dass die Analyse der Schleife

stattfindet, während die Schleife bereits (sequentiell) ausgeführt wird. Dadurch stellen wir sicher, dass die Schleifendurchführung nicht unnötig verzögert wird, falls die Schleife nicht parallelisierbar sein sollte.

- Eine Bibliothek zur einfachen Anwendung von Entwurfsmustern der parallelen Programmierung erlaubt es einem Programmierer, bekannte Strategien auszuwählen und bei ihrer Implementierung auf erprobten, effizienten Code zurückzugreifen. Wir erforschen, welche Entwurfsmuster für parallele Programmierung und insbesondere Kommunikation in parallelen Programmen existieren, und wann diese angewendet werden können. Im vergangenen Jahr haben wir insbesondere unterschiedliche Verfahren zur Wertübergabe zwischen Aktivitäten untersucht und haben bereits über 30 verschiedene Muster identifiziert. Motivation für unsere Studie ist, festzustellen welche Entwurfsmuster für Kommunikation in welchen Fällen und in welcher Umgebung mit harten Echtzeitanforderungen anwendbar ist. Ein Entwurfsmuster mit blockierenden Aufrufen, z. B. eine übliche Schlangendatenstruktur, ist für harte Echtzeit weniger passend als ein Entwurfsmuster, bei dem zumindest ein Teil der Kommunikation blockadefrei ablaufen kann. Wir werden versuchen, in einem gegebenen Anwendungsfall automatisiert eine den Anforderungen passende Implementierung eines Entwurfsmusters zu selektieren, so dass die Echtzeit-Bedingungen nicht verletzt werden.

- Unsere detaillierte Übersetzeranalyse betrachtet nicht nur einzelne Funktionen, sondern auch das gesamte Programm, um Fehler zu finden. Ein Beispiel für eine derartige detaillierte Analyse, an der wir gerade arbeiten, ist die Heap-Analyse. Diese versucht Aussagen über die Speichernutzung eines Programms zu treffen, wie z.B. ”Welche Variablen haben eine Referenz auf dieses Objekt? Welche anderen Objekte haben eine Referenz auf dieses Objekt?”. Im vergangenen Jahr haben wir ein Framework für Programmanalyse geschaffen, das die partiell inkrementelle Heap-Analyse erlaubt. Wir beabsichtigen, dieses Framework dahingehend zu erweitern, dass eine vollständig inkrementelle Heap-Analyse möglich ist.

- Bei der modellbasierten Überprüfung (Model Checker) werden alle möglichen Ausführungsreihenfolgen von Threads betrachtet, um zu bestimmen, ob ein Nebenläufigkeitsfehler auftreten kann. Die Fehler, die ein Model Checker finden kann sind u.A. Wettlaufsituationen und Verklemmungen. Ein Model Checker kann garantieren, dass ein Fehler gefunden wird. Benötigt aber viel Zeit für die Analyse des Programms. Im vergangenen Jahr haben wir Fortschritte bei der Entwicklung eines allgemeinen, sprachunabhängigen Model Checkers gemacht. Wir planen diesen mit Hilfe besserer Analysealgorithmen zu optimieren und zu parallelisieren.

- Um sicherzustellen, dass Fehler im Programmdesign möglichst früh im Entwicklungsprozess gefunden werden, ist es nötig Fehler während des Editierens des

Programms zu finden. Dazu muss die verwendete Analyse so schnell sein, dass ein interaktiver Einsatz möglich ist. Wir arbeiten an zwei Ansätzen: Unser erster Ansatz basiert darauf, die Probleme der Programmanalyse durch verbesserte Algorithmen zu lösen. Ein wichtiges Hilfsmittel ist dabei verzögerte Analyse, bei der ein Programmteil erst dann analysiert wird, wenn die Analyseergebnisse benötigt werden. Die Analyse soll inkrementell ablaufen. Dadurch wird bei kleinen Änderungen im Programmcode möglichst wenig neu analysiert. Im vergangenen Jahr haben wir unser neues Programmanalyse-Framework dahingehend erweitert, dass wir Wettlaufsituationen zum Übersetzungszeitpunkt finden, und dazu die erwähnte verzögerte und inkrementelle Analyse verwenden. Unser zweiter Ansatz basiert darauf, die Übersetzeranalysen selbst massiv zu parallelisieren um sie zu beschleunigen. Damit eine interaktive Nutzung möglich ist. Im vergangenen Jahr haben wir angefangen, grundlegende Übersetzeranalysen in eine datenparallele Darstellung zu überführen. Diese sind dann einfach auf viele verschiedene Multi-Core Architekturen portierbar. Wir haben insbesondere damit angefangen, ein generisches datenparalleles Analyse-Framework basierend auf Prädikatspropagation zu implementieren. Dieses ist auf Multi-Core CPUs als auch auf GPUs lauffähig.

2.2 InThreaT - Inter-Thread Testing

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Norbert Oster
Beginn: 1.1.2012
Kontakt:
Dr.-Ing. Norbert Oster
Tel.: +49-9131-85-28995
Fax: +49-9131-85-28809
E-Mail: norbert.oster@fau.de

Zur Beschleunigung von Rechensystemen setzen Prozessor-Hersteller schon lange nicht mehr auf steigende Taktraten - im Gegenteil: Die absolute Taktzahl sinkt, stattdessen werden in Prozessoren immer mehr unabhängige Recheneinheiten (cores) verbaut. Dafür müssen die Entwickler nun umdenken: Sie bekommen ihre Applikationen nur dann performanter (effizienter), wenn sie ihre Programme so modularisieren, dass unabhängige Codeabschnitte nebenläufig ausgeführt werden. Leider sind heutige Systeme schon funktional so komplex geworden, dass selbst die Entwicklung für eine sequentielle Ausführung noch nicht fehlerfrei gelingt - die Parallelisierung auf

mehrere Rechenkerne fügt der System-Konzeption eine weitere nicht-funktionale Komplexitätsdimension hinzu. Zwar hat die Forschung auf dem Gebiet der Softwaretechnik eine Vielzahl von Qualitätssicherungsmaßnahmen hervorgebracht, da aber die zunehmende Verbreitung von Mehrkernsystemen noch verhältnismäßig neu ist, fehlen bislang wirksame Verfahren zum Testen nebenläufiger Applikationen.

Das vorliegende Projekt hat zum Ziel, diese Lücke durch Bereitstellung eines automatisierten Testsystems zu schließen. Dazu bedarf es zunächst einer Testkriterienhierarchie, die Überdeckungsmaße speziell für das Konzept der Nebenläufigkeit bereitstellt. Vergleichbar z.B. der Verzweigungsüberdeckung für sequentielle Programme, die die Ausführung jedes Programmzweigs im Test fordert (z.B. die Bedingung eines if-statements sowohl wahr als auch falsch erzwingt - auch dann, wenn es keinen expliziten else-Zweig gibt), muss ein fundiertes Testendekriterium für nebenläufige Applikationen die systematische Ausführung aller relevanten Verschränkungen fordern (z.B. alle Reihenfolge-Kombinationen die auftreten können, wenn zwei Fäden einen gemeinsamen Speicherbereich verändern dürfen). Eine Testkriterienhierarchie schreibt dem Tester zwar vor, welche Eigenschaften seine ”fertige” Testfallmenge vorweisen muss, hilft dem Tester aber nicht bei der Identifikation der einzelnen Testfälle. Dabei genügt es nicht, wie im Falle sequentieller Tests, das Augenmerk auf die Testfälle allein zu richten: Testszenarien für parallele Module müssen zusätzlich Steuerungsinformationen zur gezielten Ausführungskontrolle der TUT (Threads Under Test) enthalten.

Im Jahr 2012 ist ein Framework für Java entstanden, das diese gezielte Ablaufsteuerung für TUT automatisch generiert. Der Tester muss lediglich die Byte-Code-Dateien seiner Applikation bereitstellen, weitere Details wie z.B. Quellcode oder Einschränkungen auf bestimmte Testszenarien kann er optional angeben, er muss sie aber nicht zwangsweise mühsam einpflegen. Der Ansatz verwendet Aspekt-Orientierte Programmierung, um die für typische Nebenläufigkeitsfehler verantwortlichen Speicherzugriffe (Lesen bzw. Schreiben von Variablen) mittels automatisch generierten Advices zu umschließen. Werden die Aspekte ins SUT (System Under Test) eingewoben, dann werden Variablenzugriffe zur Ausführungszeit abgefangen und die ausführenden Threads in der Ablaufsteuerung ”geparkt”, bis das gewünschte Testszenario erreicht ist, und dann kontrolliert in der gewünschten Reihenfolge zur weiteren Ausführung reaktiviert. Zur Demonstration wurden einige naive Ablaufsteuerungen umgesetzt, die individuelle Variablenzugriffe z.B. gezielt abwechselnd verschiedenen Threads erlauben.

2.3 OpenMP/Java

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
PD Dr. Ronald Veldema

Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Laufzeit: 1.10.2009–1.10.2015

JaMP ist eine Implementierung des bekannten OpenMP Standards für Java. JaMP erlaubt es (unter anderem) Schleifen zu parallelisieren, ohne sich mit der low-level Thread-API von Java befassen zu müssen. Eine parallele Schleife hätte in JaMP folgende Form:

```
class Test {
...void foo() {
.....//#omp parallel for
.....for (int i=0;i<N;i++) {
.........a[i]= b[i]+ c[i]
.....}
...}
}
```

JaMP implementiert im Moment die Funktionalität von OpenMP 2.0 und Teile der Spezifikation 3.0 (z.B. die collapse clause). Die aktuelle JaMP Version erzeugt reinen Java Code und ist auf jeder JVM (die Java 1.5+ unterstützt) lauffähig. Die neueste Version kann sogar CUDA fähige Hardware verwenden um Schleifen auszuführen, wenn der Schleifenrumpf eine Transformation nach CUDA möglich macht. Ist die Transformation nicht möglich, wird nebenläufiger Code für gängige Multicore Prozessoren erzeugt. JaMP unterstütz auch die gleichzeitige Nutzung von mehreren Maschinen und Acceleratoren. Dieses wurde durch die Entwicklung von zwei Abstraktionsbibliotheken ermöglicht. Die untere Abstraktionsschicht bietet abstrakte Recheneinheiten, die von den eigentlichen Berechnungseinheiten wie CPUs und GPUs und ihrem Ort in einem Rechnerbündel abstrahieren. Eine weitere Abstraktionsschicht baut auf dieser Schicht auf und bietet Operationen um partitionierte und replizierte Arrays zu verwalten. Ein partitioniertes Array wird dabei automatisch über die abstrakten Berechnungseinheiten verteilt, wobei die Geschwindigkeiten der einzelnen Berechnungseinheiten berücksichtigt werden. Welcher abstrakte Arraytyp für ein Array in einem Java-Programm konkret eingesetzt wird, wird vom JaMP-Übersetzer bestimmt, der erweitert wurde um ein Programm entsprechend zu analysieren.

Im Berichtszeitraum 2012 wurde die JaMP-Umgebung erweitert, so dass Schleifen, die Java-Objekte verwenden, ebenfalls auf Rechnerbündeln ausführbar sind. Dabei werden zwei Klassen von Objekten unterschieden. Normale Shared-Objekte werden auf

allen Berechnungseinheiten repliziert. Arrays, die Java als Objekte ansieht, können repliziert oder über die Berechnungseinheiten partitioniert werden. Dies ist unabhängig davon, ob es sich um ein Array von Objekten oder primitiven Datentypen handelt. Um die Laufzeit zu verkürzen, wurde mit der Java-Semantik gebrochen und die verzeigerte Objekt-Struktur des Programms wird nun für die Rechnerbündel auf eine flache Struktur abgebildet.

2.4 PATESIA - Parallelisierungstechniken für eingebettete Systeme in der Automatisierungstechnik

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Beginn: 1.6.2009
Förderer:
ESI-Anwendungszentrum
Mitwirkende Institutionen:
ESI-Anwendungszentrum
Kontakt:
Dipl.-Inf. Stefan Kempf
Tel.: +49-9131-85-27624
Fax: +49-9131-85-28809
E-Mail: stefan.kempf@fau.de

Dieses im Jahr 2009 gestartete Projekt befasst sich mit der Refaktorisierung und Parallelisierung von Anwendungen aus der Automatisierungstechnik. Die Programme laufen dabei auf speziellen eingebetteten Systemen. Diese Hardware bildet einen Industriestandard und kommt weltweit zum Einsatz. Da auch in eingebetteten Systemen zunehmend Multicore-Architekturen eingesetzt werden, muss bestehende, sequentielle Software für diese neuen Architekturen parallelisiert werden, um einen Zuwachs an Leistung zu gewinnen. Da die Programme typischerweise in der Industrie zur Regelung von Prozessen und zur Fertigungsautomatisierung eingesetzt werden, haben sie einen langen Lebenszyklus, um die Investitionskosten für die Unternehmen gering zu halten. Aufgrund der langen Einsatzzeit der Programme werden diese oftmals nicht mehr durch die ursprünglichen Entwickler gepflegt. Weiterhin wurde für die Programme oftmals ein hoher Aufwand betrieben, um eine zuverlässige Funktionsweise zu gewährleisten. Erweiterungen an der Software werden daher nur zögerlich vorgenommen.

Eine Migration dieser Altanwendungen auf neue Systeme und ihre anschließende Parallelisierung kann daher nicht von Hand vorgenommen werden, da sich dies als zu fehlerträchtig erweist. Es sind also Werkzeuge nötig, die diese Aufgaben automatisch vornehmen bzw. den Entwickler bei der Migration und Parallelisierung unterstützen. Für von Grund auf neu programmierte Anwendungen werden neue Programmiertechniken gebraucht. Dazu bearbeitet dieses Projekt drei Bereiche.

Der erste Bereich untersucht, welche Eigenschaften von Programmiersprachen und Techniken aus dem Übersetzerbau nötig sind, damit auch unerfahrene Entwickler effektiv echtzeitkritische Anwendungen programmieren können. Es wurden z.B. Techniken erforscht, die es einem Programmierer ermöglichen, parallele Systeme zu programmieren, ohne sich um die Synchronisation zwischen Prozessoren zu kümmern, ohne aber die Ausdrucksstärke der Programmiersprache einzuschränken. Ebenso wurde damit begonnen zu erforschen, wie derzeit notwendigen, aber zu aufwendigen Übersetzeranalysen beschleunigt werden können, so dass sie in einem interaktiven Entwicklungssystem einsetzbar sind. Die verfolgten Ansätze bestehen aus der Verwendung besserer Algorithmen sowie der Einsatz von GPUs zur Beschleunigung der Berechnungen.

Zur Parallelisierung von bestehenden Anwendungen aus der Automatisierungstechnik wurde ein spezieller Übersetzer entwickelt, um den zweiten Aspekt zu bearbeiten. Zunächst wurden Programme aus der Automatisierungstechnik auf ihre automatische Parallelisierbarkeit untersucht. Hierbei hat sich herausgestellt, dass diese zuerst refaktorisiert werden müssen, um effizient automatisch parallelisiert werden zu können. Dazu wurde als zweiter Schritt der Übersetzer so erweitert, dass er nach kritischen Programmierkonstrukten sucht, die eine Parallelisierung behindern, und den Entwickler darauf aufmerksam macht. Die zu suchenden Muster sind dabei im Übersetzer fest integriert. Aus diesem Teil entstand das bereits beschriebene Werkzeug zur Migration, bei dem sich die Muster mit einer speziellen Sprache beschreiben lassen. Die Bibliothek der Ausführungsumgebung verteilt zur Laufzeit erzeugte parallele Arbeitsaufträge dynamisch auf die verfügbaren Kerne. Dieser Mechanismus wurde durch eine Reorganisation des internen Aufbaus effizienter gestaltet, um alle Kerne gleichmäßig mit Arbeit auslasten zu können. Außerdem wurde die Bibliothek um eine Implementierung von transaktionalem Speicher sowie um ein Modell zur Interrupt-Behandlung ergänzt. Im Berichtszeitraum 2012 wurde der Übersetzer daraufhin erweitert, automatisch kritische Abschnitte in parallelen Programmen zu erkennen. Der gegenseitige Ausschluss der kritischen Abschnitte wird von der Laufzeitumgebung mittels Software-Transactional-Memory (STM) implementiert. Ferner wurden STM-Optimierungen umgesetzt. Ursprünglich verwenden STM-Systeme eine Mutex-Tabelle fester Größe zur Realisierung des gegenseitigen Ausschlusses atomarer Blöcke. Diese Konfiguration hat sich als Flaschenhals für die Programmausführung erwiesen. Die Optimierungen bestehen darin, die Tabellengröße mit der benutzten Datenmenge der Anwendung zu skalieren.

Der dritte Aspekt besteht aus der Optimierung, Refaktorisierung und Migration von Altanwendungen, um sie auf neuen Steuerungen einsatzfähig zu machen. Im Rahmen des Projekts entwickeln wir auch ein lernfähiges Expertensystem zur Unterstützung von Entwicklern in der Optimierung und Restrukturierung von Programmen. Das Ziel ist es, die zu suchenden Muster und ihre Ersetzungen nicht mehr von Hand angeben zu müssen. Stattdessen soll das System durch den Vergleich von unmodifiziertem und modifiziertem Quelltext lernen, welche Code-Transformationen durchgeführt wurden, und diese auf andere Programme anwenden können. In einem ersten Schritt wurden für mehrere Programmiersprachen (unter anderem C und Java) Zerteiler entwickelt, die Programm-Quellcode einlesen und daraus einen sprachunabhängigen abstrakten Syntaxbaum (AST) generieren. Danach wurde ein Prototyp erstellt, der Code-Konstrukte in jenen eingelesenen Syntaxbäumen findet, die sich verbessern bzw. refaktorisieren lassen. Wird ein Muster im AST gefunden, erzeugt das am Lehrstuhl entwickelte System daraus Quellcode, der entsprechende Verbesserungen enthält. Diese Quellcode-Abschnitte werden dann dem Programmierer als Vorschläge präsentiert. Zum Erzeugen der Vorschläge werden passende Muster aus einer Datenbank verwendet. Durch den Vergleich von originalem mit transformiertem Quellcode können diese Muster für die Datenbank generiert werden. Annotationen erlauben es, Muster zu generalisieren. Die Annotationen haben Zugriff auf verschiedene Plugins, um flexibel einsetzbar zu sein. Dass die so generierten Muster in verschiedenen Java- und C-Programmen gefunden werden, wurde anhand von Apache Quellcode, am Quellcode der STAMP-Benchmark-Suite und der Java-Grande Benchmark-Suite erfolgreich demonstriert. Hierbei werden Muster erkannt und dem Software-Entwickler als Vorschlag zur Transformation von (anderen) Quellcode-Abschnitten präsentiert. Im Jahr 2012 wurde das System um ein Transformationswerkzeug ergänzt, das durch die Analyse von Beispielen, bzw. den Vergleich von Versionen eines Software-Archivs, Muster generieren und in einer Datenbank speichern kann. Durch ein neues Klassifikationsverfahren können dabei irrelevante Änderungen erkannt und verworfen werden, so dass nur relevante Muster gespeichert werden.

Ein weiterer, bereits im Berichtszeitraum 2011 abgeschlossener Aspekt untersucht die Migration von Altanwendungen auf neue Systeme. Die Herausforderung hier ist, dass in den Entwicklungswerkzeugen für die neuen Plattformen nicht mehr alle Sprachkonstrukte unterstützt werden, die auf den alten Systemen eingesetzt werden konnten. Damit müssen die Programme also refaktorisiert werden. Zweitens wird untersucht, wie die Anwendungen nach einer erfolgten Migration automatisch parallelisiert werden können, wobei auch hier Anpassungen des Codes durch den Entwickler nötig sein können, um eine effiziente Parallelisierung zu gewährleisten. Damit zusammenhängend wird auch die Fragestellung der Anforderungen an ein Laufzeitsystem für parallele Programme betrachtet. In einem dritten Teilbereich wird daran geforscht, wie die ersten beiden Teilaspekte durch einen, in einem Expertensystem implementierten lernenden Ansatz kom-

biniert werden können. Ziel ist ein allgemeines Refaktorisierungswerkzeug, das sowohl für die Migration als auch für die Parallelisierung geeignet ist. Mit der Werkzeugentwicklung zur Migration wurde am Ende des Berichtszeitraums 2010 begonnen. Dieses Teilprojekt nimmt sich des Problems der automatischen Code-Refaktorisierung an. Zentrales Element ist hierbei eine eigens entwickelte Beschreibungssprache zum Auffinden von zu refaktorisierenden Code-Mustern. Damit wird es auf einfache Weise möglich, Refaktorisierungsregeln anzugeben. Ein Muster ist eine Sequenz von Anweisungen, die durch eine angepasste Code-Sequenz ersetzt werden sollen. Eine andere Möglichkeit besteht darin, den Nutzer über das Auffinden eines Musters nur zu informieren, falls die Refaktorisierung nicht automatisch durchgeführt werden kann. Die in Entwicklung befindliche Beschreibungssprache soll dabei möglichst einfach und nahe an natürlicher Sprache angelehnt sein. Ziel ist es, mit geringer Einarbeitungszeit auch komplexe Muster einfach erstellen zu können. Dazu wurde mit dem Sprachentwurf und der Einbindung des Werkzeugs in einen am Lehrstuhl entwickelten Übersetzer für Programmiersprachen der Automatisierungstechnik begonnen.

Teile des Projekts werden im Rahmen des ”ESI-Anwendungszentrums” gefördert:
http://www.esi-anwendungszentrum.de/

2.5 Übersetzerunterstützte Parallelisierung für Mehrkern-Architekturen

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Tobias Werth
Beginn: 1.3.2007
Kontakt:
Dipl.-Inf. Tobias Werth
Tel.: +49-9131-85-28865
Fax: +49-9131-85-28809
E-Mail: tobias.werth@fau.de

Die Entwicklung von schnelleren und immer effizienteren Rechnerarchitekturen ist in den letzten Jahren an verschiedene Grenzen gestoßen. Althergebrachte Techniken trugen nicht mehr oder nur noch wenig zur Beschleunigung der Hardware bei. Grundprobleme sind dabei das auseinanderdriftende Verhältnis der Latenzen von Speicher und CPU und die Abwärme bei steigenden Taktfrequenzen.

Als Lösung drängen sich homogene und heterogene Mehrkern-Architekturen auf, die dem Programmierer enorme Leistung zur Verfügung stellen. Durch verringerte Takt-

frequenzen tritt ein Großteil der genannten Problematik nicht auf, die hohe Leistung wird durch Vervielfältigung der Ressourcen erreicht. Somit sind zum Beispiel bei niedrigerem Energieverbrauch mehr Rechenoperationen pro Zeiteinheit möglich. Unter Umständen wird mittels Spezialisierung einzelner Komponenten die Rechenleistung weiter erhöht. Durch eine mehrschichtige Speicherhierarchie mit vielen Zwischenspeichern soll zum Beispiel das Problem der Latenz verkleinert werden.

Aus Mehrkern-Architekturen die volle Leistung abzurufen stellt sich als große Schwierigkeit für den Programmierer heraus. Die hohe Rechenkapazität kann er nur dann erreichen, wenn er Expertenwissen sowohl in der Domäne der Anwendung, als auch für die konkrete Architektur besitzt.

Gegenstand der Forschung sind dabei unter anderem die folgenden Fragestellungen: Welche Unterstützung kann der Übersetzer dem Programmierer beim Entwickeln von Anwendungen für verschiedenen Mehrkern-Architekturen bieten? Wie viel Kontextwissen ist notwendig, damit der Übersetzer sinnvolle Entscheidungen bei der Parallelisierung auf die einzelnen Kerne trifft? Welchen Anteil der zur Verfügung stehenden Rechenkapazität kann der Programmierer mit vertretbarem Aufwand erreichen, ohne Detailwissen über die Eigenheiten der einzelnen Architekturen besitzen zu müssen? Wie müssen geeignete Werkzeuge zum Auffinden von Fehlern und Flaschenhälsen in der Anwendung auf Mehrkern-Architekturen aussehen?

Ziel dieses Projektes ist es, diese Fragen anhand einer eingeschränkten Anwendungsdomäne zu beantworten und mögliche Lösungswege aufzuzeigen. Als Domäne wird das Lattice-Boltzmann-Verfahren herangezogen, das vor allem in der Strömungssimulation angewandt wird. Durch seine Gitterstruktur und eine überschaubare Anzahl an Datenabhängigkeiten zwischen den einzelnen Zellen lässt sich das Verfahren relativ einfach parallelisieren, so dass sich die Forschung auf die oben genannten Fragestellungen konzentrieren kann.

Die heterogene CellBE-Architektur bietet sich aufgrund ihrer enormen Leistung auf einem Chip als Zielarchitektur an. Sie besteht aus einem PowerPC-Kern (PPU) und acht sogenannten Synergistic Processing Units (SPUs), die Berechnungen parallel ausführen können. Das Programmiermodell Cilk für die CellBE-Architektur wurde weiterentwickelt, um eine stabile und effiziente Ausführung auf den SPUs zu ermöglichen. Dabei wurden vor allem Möglichkeiten geschaffen, die das Finden von Fehlern beim Stehlen von Funktionen auf entfernten SPUs erleichtern. Außerdem wurde die Quell-zu-Quellcode-Transformation überarbeitet, um leichter den entsprechenden Code für PPU und SPU erzeugen zu können.

Im Jahr 2012 wurden Grafikkarten (GPUs) als weitere Zielarchitektur ins Auge gefasst, die heutzutage eine weitaus höhere Rechenleistung im Gegensatz zu normalen Prozessoren anbieten. Diese Rechenleistung kann durch die Programmierung mit Cuda (NVidia) oder OpenCL (AMD) aufgrund ihrer Struktur für datenparallele Probleme

relativ einfach abgerufen werden. Weitaus schwieriger ist die Umsetzung taskparalleler Anwendungen, die im weiteren Verlauf des Projekts untersucht werden sollen. Dazu sollen verschiedene Lastausgleichsalgorithmen entworfen, prototypisch umgesetzt und miteinander verglichen werden. Im Jahr 2012 wurde ein erstes Verfahren mit mehrstufigen Warteschlangen und dem Prinzip der Arbeitsspende (work donation) entworfen.

2.6 Effiziente Software-Architekturen für verteilte Ereignisverarbeitungssysteme

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Christopher Mutschler
Laufzeit: 15.11.2010–14.11.2013
Förderer:
Fraunhofer Institut für Integrierte Schaltungen

Funkortungssysteme, auch bekannt als Real-Time Location Systems (RTLS), geraten immer mehr in den Fokus der Logistik, Produktion und vieler weiterer Prozesse. Diese Systeme liefern wertvolle Informationen über den Aufenthaltsort von beteiligten Objekten zur Laufzeit. Damit können Prozesse verfolgt, analysiert und optimiert werden. Neben den Forschungsbereichen an der Basis von Ortungssystemen, wie robuste und störsichere Ortungstechnologien oder Verfahren zur hochgenauen Positionsbestimmung, rücken mehr und mehr Methoden in den Vordergrund, die aus Positionsdatenströmen wertvolle Informationen für weitere Verarbeitungsstufen gewinnen. In diesem Kontext erforscht das Projekt Verfahren zur Ereignisdetektion in Positionsdatenströmen zur Laufzeit.

2011 wurde damit begonnen, auftretende Ereignisse in Lokalisierungssystemen zu erkennen und vorherzusagen. Hierfür werden Ereignisströme zur Laufzeit analysiert und ausgewertet. Somit konnten Modelle erlernt werden, um Ereignisse aus Ereignisströmen zu prädizieren.

2012 wurden für die Laufzeitanalyse von Positionsdatenströmen mehrerer Methoden entwickelt, um Ereignisse mit möglichst geringer Latenz detektieren zu können. Hierbei können einzelne Teilereignisse durch sog. Ereignisdetektoren dazu verwendet werden, höhere Zusammenhänge in den Daten hierarchisch zusammenzusetzen. Hierdurch wird die Komplexität der einzelnen Detektionskomponenten drastisch reduziert. Diese werden somit wartbarer und durch die Ausnutzung paralleler und verteilter Rechnerstrukturen wesentlich effizienter. Es ist nun möglich, Ereignisse in den Positionsdatenströmen innerhalb von nur einigen hundert Millisekunden zu erkennen.

Das Projekt stellt einen Beitrag des Lehrstuhls Informatik 2 zum IZ ESI dar.

2.7 Softwareleitstand - Prototypische Entwicklung eines neuartigen Werkzeugs zur Qualitätsabsicherung bei der Softwareentwicklung

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Josef Adersberger
Norbert Tausch, M. Eng.
Laufzeit: 1.11.2009–31.12.2013
Förderer:
Bundesministerium für Wirtschaft und Technologie
Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49-9131-85-27625
Fax: +49-9131-85-28809
E-Mail: michael.philippsen@fau.de

Moderne Softwaresysteme werden sowohl fachlich, technisch als auch organisatorisch zunehmend komplexer: So steigt die Anzahl und der Vernetzungsgrad der zu realisierenden Anforderungen pro System stetig, die technischen Vorgaben z.B. an den Verteilungsgrad und die Zuverlässigkeit der Systeme werden komplexer und die Softwareentwicklung selbst findet zunehmend in global verteilten Teams und mit wachsendem Zeitdruck statt. Aus diesen Gründen wird es auch zunehmend schwieriger, Softwareentwicklungsprojekte fachlich, technisch und organisatorisch zu steuern.

Als Softwareleitstand bezeichnen wir ein Werkzeug, das leitenden Projektrollen wie dem Projektleiter, dem Softwarearchitekten, dem Anforderungsarchitekten und dem Entwicklungsleiter eine hohe Transparenz und damit verbesserte Steuerbarkeit von Softwareentwicklungsprojekten ermöglicht.

Transparenz herrscht dann, wenn sowohl Zusammenhänge zwischen den vielerlei Erzeugnissen eines Softwareentwicklungsprojekts als auch deren Eigenschaften schnell und gesamtheitlich zugänglich sind und entsprechend dem individuellen Informationsbedarf eines Projektbeteiligten aufbereitet sind.

Der Softwareleitstand ist ein Werkzeug, das den Zugang zu den Zusammenhängen (Traceability) und den Eigenschaften (Metriken) der Erzeugnisse von Softwareentwicklungsprojekten vereinheitlicht. Damit kann die Effizienz von Softwareentwicklungsprojekten maßgeblich gesteigert werden. Es sollen Erzeugnisse des Softwareentwicklungsprojekts (Artefakte) und ihre Zusammenhänge (Relationen), sowie zu den Artefakten zuordenbare Metriken zentral erfasst, integriert und analysiert werden können. Die ent-

sprechenden Analysen werden in Form von Visualisierungen des Artefaktgraphen mitsamt den zugeordneten Metriken und Regelprüfungen durchgeführt.

Das Projekt Softwareleitstand wird in Kooperation des Lehrstuhls mit der QAware GmbH München durchgeführt und wurde von 2009 bis 2012 aus Mitteln des BMWi gefördert. Die Projektlaufzeit ist November 2009 bis Dezember 2013. Die Umsetzung des Softwareleitstands erfolgt dabei in zwei Arbeitssträngen, die auch den beiden Subsystemen des Werkzeugs entsprechen: Der Integration Pipeline, die Traceability Informationen und Metriken aus verschiedensten Werkzeugen der Softwareentwicklung zusammen sammelt sowie dem Analysis Core (Analysekern), der eine gesamtheitliche Auswertung der integrierten Daten ermöglicht.

Die Integration Pipeline wird durch den Projektpartner QAware GmbH entwickelt. Dabei wurde im bisherigen Projektverlauf zunächst eine Modellierungssprache für Traceability Informationen in Kombination mit Metriken (TraceML) definiert. Die Sprache besteht dabei aus einem Meta-Modell sowie einer Modellbibliothek zur einfachen Definition von angepassten Traceability Modellen. Aufbauend auf der TraceML wurde das Integration Pipeline Framework auf Basis des Eclipse Modeling Projekts entwickelt. Dabei wird sowohl das Eclipse Modeling Framework zur Abbildung der Modelle und Metamodelle, als auch die Modeling Workflow Engine zur Modelltransformation und Eclipse CDO als Modell-Repository verwendet. Auf Basis des Integration Pipeline Frameworks wurden dann eine Reihe von gängigen Werkzeugen der Softwareentwicklung wie z.B. Subversion, Eclipse, JIRA, Enterprise Architect und Maven angebunden.

Der Analysekern wird durch den Lehrstuhl entwickelt. Zentrales Thema ist dabei die Konzeption und Realisierung einer domänenspezifischen Sprache für die graph-basierte Traceability-Analyse. Das Ziel dieser Traceability Query Language (TracQL) ist es den Aufwand zur Umsetzung von Traceability-Analysen zu reduzieren. Dies ist derzeit der Hauptkritikpunkt von Industrie und Wissenschaft, der die Umsetzung der Traceability hemmt. TracQL erleichtert dabei sowohl die Extraktion als auch die Transformation der Traceability-Daten, so dass diese dann mittels kurzer funktional formulierter Graph-Traversierungen analysiert werden können. Die Sprache baut auf der mutli-paradigmen Sprache Scala auf und wurde im Berichtszeitraum bereits mehrfach in realen Industrieprojekten zur Analyse erfolgreich eingesetzt.

2.8 Integrierte Werkzeug-Kette zur metamodellbasierten Modellierung und Ausführung von Software-Entwicklungsprozessen

Projektleitung:
Hon.-Prof. Dr.-Ing. Detlef Kips
Beteiligte:
Dipl.-Inf. Ralf Ellner

Prof. Dr. Michael Philippsen
Dr.-Ing. Martin Jung
Dipl.-Inf. Johannes Drexler
Dipl.-Inf. Samir Al-Hilank
Laufzeit: 1.10.2008–31.12.2012
Förderer:
BMWi

Aufgrund ständig wachsender Anforderungen, die an die Entwicklung komplexer Softwaresysteme gestellt werden, gewinnt die Einhaltung wohldefinierter Software-Entwicklungsprozesse (SWEPe) immer mehr an Bedeutung. Im Kontext umfangreicher, global verteilter Entwicklungsprojekte ist dabei insbesondere ein Trend zu organisationsübergreifenden, langlaufenden und dabei dynamisch veränderbaren Prozessen erkennbar. Zur effektiven Beschreibung und Unterstützung solcher Entwicklungsprozesse sind speziell geeignete Prozessmodellierungssprachen und eine mächtige Werkzeugunterstützung unverzichtbar.

Die Ergebnisse vorangegangener Untersuchungen machten deutlich, dass der Markt für SWEP-Beschreibungs- und -Ausführungsumgebungen derzeit noch keine Lösungen bietet, die eine hinreichend präzise und flexible Modellierung von Entwicklungsprozessen sowie deren automatisierte Ausführung, Steuerung und Überwachung ermöglichen. Diese Lücke wurde im Rahmen eines Kooperationsprojektes geschlossen, das in Zusammenarbeit mit der develop group als Industriepartner durchgeführt und mit Mitteln des BMWi gefördert wurde. Es wurde im Oktober 2008 mit drei wissenschaftlichen Mitarbeitern gestartet und im September 2011 abgeschlossen.

Ziel dieses Kooperationsprojektes war es, auf Grundlage eines durchgängigen, metamodellbasierten Ansatzes eine integrierte Werkzeugkette für die Modellierung und Ausführung industrieller Software-Entwicklungsprozesse prototypisch zu realisieren. Im Hinblick auf die Praxistauglichkeit der Lösung lag das Hauptaugenmerk dabei auf der Anpassbarkeit der Prozessmodelle an verschiedene industrielle Entwicklungsszenarien, auf der Anwenderfreundlichkeit der Prozessbeschreibung und auf einer weitgehenden Automatisierung der Prozessausführung, die zur Effizienzsteigerung in der Entwicklung entscheidend beiträgt. Diese charakteristischen Vorzüge werden durch einen relativ hohen Formalisierungsgrad der Prozessmodellierung, durch eine weitgehende Generizität der Modellierungs- und Prozessausführungswerkzeuge sowie durch die Verwendung verbreiteter und akzeptierter Industriestandards (UML, SPEM) erreicht.

Als Basis für die integrierte Werkzeugkette kommt eine Erweiterung des SPEM-Standards (eSPEM – enactable SPEM) zum Einsatz. eSPEM erweitert das SPEM um Konzepte zur Verhaltensmodellierung auf Grundlage der UML-Aktivitäts- und -Zustandsmaschinendiagramme. Als Ergänzung bietet das eSPEM spezifische Sprachkonstrukte, um bestimmte für SWEPe typische Sachverhalte angemessen modellieren

zu können, darunter u.a. die dynamische Erzeugung und Planung von Entwicklungsaktivitäten und Arbeitsschritten.

Im Berichtszeitraum 2012 wurde eine Zusammenfassung der integrierten Werkzeugkette und des eSPEM anlässlich des Workshops ”First Workshop on Academics Modeling with Eclipse” auf der ”8th European Conference on Modelling Foundations and Applications” veröffentlicht. Bei der Definition von SWEPen im industriellen Kontext wurde außerdem deutlich, dass Referenzmodelle und Normen einen zunehmend starken Einfluss auf die Modellierung von SWEPen ausüben. Im Mittelpunkt solcher Normen und Referenzmodell, die nachfolgend als Qualitätsstandards bezeichnet werden, stehen dabei oft Verfahren und Vorgehensweisen (sogenannte Best Practices), die auf eine Verbesserung der Qualität des zu entwickelten SW-Produktes abzielen oder eine gesteigerte Effizienz des Entwicklungsvorhabens zum Ziel haben (z.B. CMMI oder Automotive SPICE). Andere Qualitätsstandards, wie z.B. die ISO 26262 Functional Safety – Road Vehicles, stellen Anforderungen an die im SWEP definierten Maßnahmen zur Entwicklung eines sicheren SW-Produktes (Safety). In diesem Zusammenhang ist ein weiteres Ziel des Forschungsprojektes, diese Anforderungen aus Qualitätsstandards mit dem SWEP zu verknüpfen. Damit soll die effiziente Durchführung von Assessments des SWEPes und Prozessverbesserungsprojekten unterstützt werden. Der Schwerpunkt liegt dabei insbesondere auf Szenarien, in denen mehr als ein Qualitätsstandard zum Einsatz kommt (z.B. CMMI, Automotive SPICE und ISO 26262).

2.9 Embedded Systems Institute

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Christopher Mutschler
Andreas Kumlehn, M. Sc.
Dr.-Ing. Norbert Oster
Beginn: 1.9.2007

Das im September 2007 als Interdisziplinäres Zentrum an der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) gegründete ”ESI - Embedded Systems Institute” hat sich die fächerübergreifende Koordination und Organisation der Forschung, Lehre und Weiterbildung im Bereich Eingebetteter Systeme zum Ziel gesetzt.

Über das ESI werden an der Universität vorhandene Kompetenzen mit den Interessen, Aktivitäten und Zielen der einschlägigen Großindustrie und des Mittelstands auf dem Gebiet des Entwurfs Eingebetteter Systeme vernetzt.

Unternehmen erhalten durch das ESI Zugriff auf neueste Forschungsergebnisse sowie die Möglichkeit, gemeinsam Entwicklungsprojekte durchzuführen, Kontakte zu knüpfen und Kooperationspartner zu finden. Das ESI bündelt die Kompetenzen der Lehrstühle und macht sie für Kooperationsprojekte nutzbar. Aktuelle Forschung lässt sich damit schneller in Produkte umsetzen. Beschleunigt wird auch der Aufbau gemeinsamer Forschung. Schließlich dient das ESI auch als Schnittstelle zum frühzeitigen Zugriff auf Studierende und fachlich qualifizierten Nachwuchs.

Der Lehrstuhl Informatik 2 (Prof. Dr. Michael Philippsen) gehört zu den aktiv beteiligten Gründungsmitgliedern des ESI und führt in diesem Rahmen Forschungsprojekte durch.

Weitere Informationen finden Sie auch unter http://www.esi.uni-erlangen.de sowie http://www.esi-anwendungszentrum.de

2.10 Graphen und Graphtransformationen

Projektleitung:
Prof. em. Dr. Hans Jürgen Schneider
Beginn: 1.10.2004
Kontakt:
Prof. em. Dr. Hans Jürgen Schneider
Tel.: +49-9131-85-27620
Fax: +49-9131-85-28809
E-Mail: hans.juergen.schneider@fau.de

Graphen werden an vielen Stellen als intuitives Hilfsmittel zur Verdeutlichung komplizierter Sachverhalte verwendet. Außerhalb der Informatik trifft dies z.B. auf die Chemie zu, wo Moleküle graphisch modelliert werden. Innerhalb der Informatik werden beispielsweise Daten- bzw. Kontrollflussdiagramme, Entity-Relationship-Diagramme oder Petri-Netze zur Visualisierung sowohl von Software- als auch von Hardware-Architekturen verwendet. Graphgrammatiken und Graphtransformationen kombinieren Ideen aus den Bereichen Graphentheorie, Algebra, Logik und Kategorientheorie, um Veränderungen an Graphen formal zu beschreiben.

Die Kategorientheorie ist ein attraktives Hilfsmittel, äußerst unterschiedliche Strukturen in einer einheitlichen Weise zu beschreiben, z.B. die unterschiedlichen Modelle für asynchrone Prozesse: Petri-Netze basieren auf gewöhnlichen markierten Graphen,

Statecharts verwenden hierarchische Graphen, die parallele logische Programmierung kann mit Hilfe sogenannter Dschungel graphentheoretisch interpretiert werden, und die Aktorsysteme lassen sich als Graphen darstellen, deren Markierungsalphabet eine Menge von Termgraphen ist.

Im Jahre 2012 haben wir uns auf einen theoretischen Aspekt konzentriert.

Unsere Arbeiten zu den Graphtransformationen basieren auf Konzepten der Kategorientheorie. Der sogenannte Doppelpushout-Ansatz stellt eine Produktion durch zwei Morphismen dar, die an einem gemeinsamen Interface-Graphen ansetzen. Das eine Pushout fägt die linke Seite der Produktion in den Kontext ein, das andere die rechte Seite. Wenn wir einen Ableitungsschritt tatsächlich konstruieren wollen, müssen wir auf der linken Seite ein Pushout-Komplement finden, was als nachteilig empfunden wird. Raoult hat 1984 vorgeschlagen, die Graphersetzung durch ein einzelnes Pushout zu beschreiben; der Ansatz wurde dann von Loewe ausführlich diskutiert, wobei die Untersuchungen weitgehend auf injektive Morphismen beschränkt blieben. Unter dieser Voraussetzung sind die Ansätze äquivalent. Jedoch führen einige Anwendungen, z.B. die Termersetzung, zu nichtinjektiven Morphismen. Wir haben diese Fälle detailliert untersucht und konnten zeigen, dass sie auch in diesem Fall äquivalent sind, solange die Einbettung vernünftige Eigenschaften hat.

2.11 International Collegiate Programming Contest an der FAU

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Daniel Brinkers
Dipl.-Math. Jakob Krainz
Beginn: 1.11.2002
Kontakt:
Dipl.-Inf. Tobias Werth
Tel.: +49-9131-85-28865
Fax: +49-9131-85-28809
E-Mail: tobias.werth@fau.de

Die Association for Computing Machinery (ACM) richtet seit Jahrzehnten den International Collegiate Programming Contest (ICPC) aus. Dabei sollen Teams aus je drei Studenten in fünf Stunden neun bis elf Programmieraufgaben lösen. Als Erschwernis kommt hinzu, dass nur ein Computer pro Gruppe zur Verfügung steht. Die Aufgaben erfordern solide Kenntnisse von Algorithmen aus allen Gebieten der Informatik und Mathematik, wie z.B. Graphen, Kombinatorik, Zeichenketten, Algebra und Geometrie.

Der ICPC wird jedes Jahr in drei Stufen ausgetragen. Zuerst werden innerhalb der Universitäten in lokalen Ausscheidungen die maximal drei Teams bestimmt, die dann zu den regionalen Wettbewerben entsandt werden. Erlangen liegt seit dem Jahr 2009 im Einzugsbereich des Northwestern European Regional Contest (NWERC), an dem u.a. auch Teams aus Großbritannien, den Benelux-Staaten und Skandinavien teilnehmen.

Die Sieger aller regionalen Wettbewerbe der Welt (und einige Zweitplatzierte) erreichen die World Finals, die im Frühjahr des jeweils darauffolgenden Jahres (2013 in St. Petersburg, Russland) stattfinden. Im Jahr 2012 fanden zwei lokale Wettbewerbe an der FAU statt. Im Wintersemester wurde mit einer Rekordzahl von 25 Teams ein Mannschaftswettbewerb mit dem Ziel ausgetragen, neue Studierende für die Wettbewerbe zu begeistern. Jedes Team bestand aus maximal drei Studenten. Außerdem nahmen noch 40 Teams von anderen europäischen Universitäten teil.

Im Sommersemester fand zum wiederholten Mal das Hauptseminar "Hallo Welt! - Programmieren für Fortgeschrittene" statt, um Studierende verschiedener Fachrichtungen in Algorithmen und Wettbewerbsaufgaben zu schulen. Der Wettbewerb im Sommersemester diente der Auswahl der studentischen Vertreter der FAU für den NWERC 2012. Insgesamt nahmen an dem deutschlandweit organisierten Ausscheidungskampf 21 Teams der FAU mit Studenten verschiedensten Fachrichtungen teil - das Team "easy cheesy peasy" aus Erlangen konnte den Wettbewerb gewinnen (dritter Sieg in Folge). Aus den besten Teams wurden neun Studenten ausgewählt, die für den NWERC Dreierteams bildeten (ein zehnter Student wurde als Ersatzmann ausgewählt). Das beste Team der FAU löste im November 2012 beim nordwesteuropäischen Wettbewerb NWERC in Delft sechs Aufgaben und verpassten damit knapp die Medaillenränge. Das zweite Team konnte fünf Aufgaben lösen und landete auf dem 22. Platz von 83 Teams. Der Erfolg wurde durch das dritte Team abgerundet, das im Mittelfeld Platz 38 erreichte.

3 Publikationen 2012

– Braunstein, Sören ; Härdtlein, Jochen ; Philippsen, Michael: Expressing Parallelism and Timing in Embedded Real-Time Applications . In: High-Performance and Embedded Architecture and Compilation (HiPEAC) Network of Excellence (Veranst.) : 8th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES) 2012 - Poster Abstracts (8th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems

Fiuggi, Italy July 11, 2012). Ghent (Belgium) : Academia Press, 2012, S. 197-200. - ISBN 978-90-382-1987-5

– Dotzler, Georg ; Veldema, Ronald ; Philippsen, Michael: Annotation Support for Generic Patches . In: Maalej, Walid ; Robillard, Martin ; Walker, Robert J. ; Zimmermann, Thomas (Hrsg.) : Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering (RSSE 12) (International Workshop on Recommendation Systems for Software Engineering Zurich, Switzerland 04.06.2012). 2012, S. 6-10. - ISBN 978-1-4673-1758-0

– Ellner, Ralf ; Al-Hilank, Samir ; Jung, Martin ; Kips, Detlef ; Philippsen, Michael: An Integrated Tool Chain for Software Process Modeling and Execution . In: Störrle, Harald ; Botterweck, Goetz ; Bourdellès, Michel ; Kolovos, Dimitris; Paige, Richard ; Roubtsova, Ella ; Rubin, Julia ; Tolvanen, Juha-Pekka (Hrsg.): Joint Proceedings of co-located Events at the 8th European Conference on Modeling Foundations and Applications (ECMFA 2012) (8th European Conference on Modeling Foundations and Applications (ECMFA 2012) Lyngby, Denmark 02.-05.07.2012). 2012, S. 73-82. - ISBN 978-87-643-1014-6

– Mutschler, Christopher ; Philippsen, Michael: Apparatus, Method and Computer Program for Migrating an Event Detector Process . Schutzrecht PCT/EP2011/069159 Patentanmeldung (14.03.2012)

– Mutschler, Christopher ; Philippsen, Michael: Learning Event Detection Rules with Noise Hidden Markov Models . In: Benkrid, Khaled ; Merodio, David (Hrsg.) : Proceedings of the 2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2012) (2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2012) Nuremberg, Germany 25.06.2012). 2012, S. 159-166. - ISBN 978-1-4673-1914-0

– Mutschler, Christopher ; Philippsen, Michael: Towards a Distributed Self-Optimizing Event Processing System for Realtime Locating Systems (RTLS) . In: ACM (Hrsg.) : DEBS PhD Workshops, 6th ACM International Conference on Distributed Event-Based Systems (International Conference on Distributed Event-Based Systems (DEBS) Berlin, Germany July 16-20, 2012). 2012, S. -.

– Mutschler, Christopher ; Loeffler, Christoffer: Vorrichtung, Verfahren und Computerprogramm zum Verbessern einer Leistungsfähigkeit eines ereignisbasierten verteilten Analysesystems . Schutzrecht DE 10 2012 112 253.9 Patentanmeldung (13.12.2012)

– Otto, Stephan ; Bretz, Ingmar ; Franke, Norbert ; von der Grün, Thomas ; Mutschler, Christopher: Vorrichtung, Verfahren und Computerprogramm zur Rekon-

struktion einer Bewegung eines Objekts . Schutzrecht 10 2012 111 304.1 DE Patentanmeldung (22.11.2012)

– Pankratius, Victor ; Philippsen, Michael (Hrsg.): Multicore Software Engineering, Performance and Tools (Proceedings MSEPT 2012) . (International Conference on Multicore Software Engineering, Performance, and Tools (MSEPT 2012) Prague, Czech Republic 31.05. - 01.06.2012) Berlin Heidelberg : Springer, 2012 (Lecture Notes in Computer Science (LNCS) Bd. 7303) . - 95 Seiten. ISBN 978-3-642-31201-4. ISSN 0302-9743

– Tausch, Norbert ; Philippsen, Michael ; Adersberger, Josef: TracQL: A Domain-Specific Language for Traceability Analysis . In: Ali Babar, M. ; Cuesta, C. ; Savolainen, J. ; Männistö, T. (Hrsg.) : Proceedings of the 2012 Joint Working Conference on Software Architecture & 6th European Conference on Software Architecture (Joint Working Conference on Software Architecture & 6th European Conference on Software Architecture (WICSA/ECSA 2012) Helsinki, Finland 20. - 24.08.2012). Los Alamitos, CA : IEEE CPS, 2012, S. 320-325. - ISBN 978-0-7695-4827-2

– Veldema, Ronald ; Philippsen, Michael: Parallel Memory Defragmentation on a GPU . In: Zhang, Lixin ; Mutlu, Onur (Hrsg.) : Proceedings of the 2012 ACM SIGPLAN Workshop on Memory Systems Performance and Correctness (ACM SIGPLAN Workshop on Memory Systems Performance and Correctness (MSPC 12) Beijing, China 16.06.2012). 2012, S. 38-47. - ISBN 978-1-4503-1219-6

4 Studien- und Abschlussarbeiten

– Studienarbeit: Realisierung der konkreten Syntax von ”DeepML” mit dem Graphiti-Framework. Bearbeiter: Elena Klevtsova (beendet am 01.06.2012); Betreuer: Dipl.-Inf. Ralf Ellner; Hon.-Prof. Dr.-Ing. Detlef Kips

– Master Thesis: Blending Agile and Traditional Project Management for Safety Critical Systems Engineering. Bearbeiter: Tobias Freitag (beendet am 09.08.2012); Betreuer: Hon.-Prof. Dr.-Ing. Bernd Hindel

– Bachelor Thesis: Programmierung einer Mini-Anlage mit parallelen Programmierkonstrukten. Bearbeiter: Martin Sturm (beendet am 28.08.2012); Betreuer: PD Dr. Ronald Veldema; Dipl.-Inf. Stefan Kempf; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Anpassung eines Expertensystems zur Programmanalyse und -refaktorisierung für Anwendungen aus der Automatisierungstechnik. Bearbeiter:

Johannes Wellhöfer (beendet am 03.09.2012); Betreuer: Dipl.-Inf. Georg Dotzler; Dipl.-Inf. Stefan Kempf; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Evaluation von Algorithmen zur Positionsdatenkompression. Bearbeiter: Jonathan Reuss (beendet am 2.10.2012); Betreuer: Dipl.-Inf. Christopher Mutschler; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Implementierung eines JaMP-Übersetzers zur Generierung von Java/OpenCL Code. Bearbeiter: Florian Habur (beendet am 02.10.2012); Betreuer: Dipl.-Inf. Georg Dotzler; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Netzwerksimulation- und -optimierung von verteilten Ereignisverarbeitungssystemen. Bearbeiter: Löffler Christoffer (beendet am 05.11.2012); Betreuer: Dipl.-Inf. Christopher Mutschler; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Entwicklung eines Werkzeugs zur Extraktion von Mustern aus Software-Archiven zur Quellcode-Optimierung. Bearbeiter: Marius Kamp (beendet am 06.11.2012); Betreuer: Dipl.-Inf. Georg Dotzler; Prof. Dr. Michael Philippsen

– Diplomarbeit: Konzeption und prototypische Realisierung einer Abfrage- und Transformationssprache für DeepML. Bearbeiter: Elena Klevtsova (beendet am 09.11.2012); Betreuer: Dipl.-Inf. Samir Al-Hilank; Dipl.-Inf. Ralf Ellner; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Intelligent Documentation Generation. Bearbeiter: Teodor Shaterov (beendet am 27.11.2012); Betreuer: PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen

– Master Thesis: Transparent use of Java objects on the GPU in the JaMP/OpenMP framework. Bearbeiter: Carolin Wolf (beendet am 06.12.2012); Betreuer: Dipl.-Inf. Georg Dotzler; Prof. Dr. Michael Philippsen