Jahresbericht 2010
Lehrstuhl 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
Honorar- und Außerplanmäßige Professoren:
Hon.-Prof. Dr.-Ing. Bernd Hindel
Hon.-Prof. Dr.-Ing. Detlef Kips
apl. Prof. Dr.-Ing. Gabriella K'okai, †06.04.2010
Emeritus:
Prof. em. Dr. Hans Jürgen Schneider
Sekretariat:
Agnes Brütting
Margit Zenk
Wiss. Mitarbeiter:
Dipl.-Inf. Thorsten Blaß (ab 15.10.2010)
Dipl.-Inf. Daniel Brinkers (ab 10.01.2011)
Dipl.-Inf. Georg Dotzler (ab 01.02.2010)
Dipl.-Inf. Alexander Dreweke (bis 31.01.2011)
Dipl.-Ing. (FH) Thorsten Edelhäußer
Dipl.-Inf. Ralf Ellner
Dipl.-Inf. Philipp Janda
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Christopher Mutschler (ab 15.11.2010)
Dr.-Ing. Norbert Oster
M. Eng. Norbert Tausch, Dipl.-Ing. (FH)
PD Ronald Veldema, Ph.D.
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Marc Wörlein
Gast:
Dipl.-Inf. (FH) Josef Adersberger
Dr.-Ing. Martin Jung
Dr. Karsten Schmidt
Externes Lehrpersonal:
Dr.-Ing. Klaudia Dussa-Zieger

Dipl.-Inf. Stephan Otto

Der 1972 gegründete Lehrstuhl Informatik 2 (Programmiersyteme) wird seit April 2002 von Prof. Dr. Michael Philippsen (als Nachfolger von Prof. Dr. 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 Software 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 Übersetzerunterstützte Parallelisierung für Mehrkern-Architekturen

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Dominic Schell
Beginn: 1.3.2007
Kontakt:
Dipl.-Inf. Tobias Werth
Tel.: +49 9131 85-28865
Fax: +49 9131 85-28809
E-Mail: tobias.werth@informatik.uni-erlangen.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 Taktfrequenzen 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.

Im Jahr 2010 wurde folgende Teilaufgaben bearbeitet:

- 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.

- Da OpenCL eine gute Grundlage bietet, neben der CellBE-Architektur auch andere Plattformen zu unterstützen, wurde 2010 begonnen, das Lattice-Boltzmann-Verfahren mittels OpenCL auf verschiedenen Architekturen zu testen. Dabei wurde erforscht, welche Details der vorhandenen Implementierungen der OpenCL-Spezifikation und der darunterliegenden Hardware von OpenCL abstrahiert werden können ohne an Effizienz zu verlieren.

- Desweiteren wurde eine freie Bibliothek zur Simulation nach dem Lattice-Boltzmann-Verfahren (OpenLB bzw. PaLaBoS) untersucht. Dabei wurden verschiedene Ansätze für die Parallelisierung des Verfahrens auf heterogene Architekturen prototypisch implementiert und evaluiert. Wir haben ferner begonnen, PaLaBoS so zu erweitern, dass für verschiedenen Modelle des Lattice-Boltzmann-Verfahrens automatisiert OpenCL-Code erzeugt wird, um die Simulation auf der Grafikkarte ablaufen zu lassen.

2.2 Graphbasierte Prozedurale Abstraktion

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Alexander Dreweke
Dipl.-Inf. Marc Wörlein
Dipl.-Inf. Tobias Werth
Laufzeit: 1.4.2006–31.12.2010
Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49 9131 85-27625
Fax: +49 9131 85-28809
E-Mail: philippsen@informatik.uni-erlangen.de

Als besonders dringend erscheint uns gegenwärtig die Verbesserung der Programmierwerkzeuge für eingebettete Systeme. Solche Systeme werden heutzutage zu oft noch sehr maschinennah programmiert. Das inzwischen bei der Programmierung von Arbeitsplatzrechnern übliche Abstraktions- und Komfortniveau (Objektorientierung, automatische Speicherbereinigung, Ausnahmebehandlung, Parallelität, Aspektorientierung, Komponenten, ...) scheint im Bereich der eingebetteten Systeme noch in weiter Ferne, wodurch Portabilität, Robustheit und Wartbarkeit der erstellten Software erheblich beeinträchtigt wird. Dies ist ein erhebliches volkswirtschaftliches Problem,

das gerade auch deshalb bedeutsam ist, weil Europa auf diesem Feld nicht von Amerika dominiert wird. Fernziel muss es daher sein, das Abstraktionsniveau bei der Programmierung eingebetteter Systeme schrittweise zu erhöhen, indem Optimierungstechniken entwickelt werden, die trotz des erhöhten Abstraktionsniveaus "kleinen" Code garantieren.

Neben der offensichtlichen Frage, wie die bekannten Optimierungstechniken auf die Code-Größe wirken, drängen sich neue Einzelfragen auf. Während der RAM-Bedarf einer Applikation auf Desktop-Rechnern kaum eine Rolle spielt, ist dieser für eingebettete Systeme oft essentiell. Objektorientierter - vor allem bibliotheksbasierter - Code bietet ein erhebliches, bislang ungenutztes Potential für prozedurale Abstraktion zur Code-Verkleinerung. Neben der Code-Größe kommt auch dem Aspekt der Energie-Effizienz eine wachsende Bedeutung als Zielgröße der Code-Optimierung zu. Hier muss der Übersetzer, ggf. im Zusammenspiel mit dem Betriebssystem, optimieren bzw. auf die Hardware-Parameter einwirken. Die Behandlung der nicht-uniformen Speicherzugriffshierarchie, die in verteilten Systemen neben Registern, Cache und Hauptspeicher um eine weitere Leistungsebene vertieft ist, stellt auch bei eingebetteten Systemen eine Herausforderung dar, da z.B. Flash-Speicher zu berücksichtigen sind. Können eingebettete Systeme (ebenso verteilte Systeme) - der Tradition von Desktop-Prozessoren folgend - auch weiterhin mit der Illusion eines transparenten Zugriffs programmiert werden? Kann man durch statische Analyse Informationen über bestehende Lokalitätsbeziehungen zwischen Daten extrahieren? Welche Optimierungen sind dann möglich? Profitieren statische Analyse und Laufzeitmechanismen von einander? Wie können durch Programmanalyse Pre-Fetch- und Post-Store-Kommandos im generierten Code erzeugt werden, durch die Cache-Effekte überdeckt, Wartezeiten vermieden oder Energie gespart werden?

Eine gängige Methode zur Code-Größenverkleinerung ist die Prozedurale Abstraktion (PA): gleiche Code-Abschnitte im Programm werden gesucht und daraus eine einzige neue Prozedur erzeugt. Die Code-Abschnitte werden durch Aufrufe der neu erzeugten Prozedur ersetzt, wodurch die Redundanz innerhalb eines Programms und somit seine Größe reduziert wird. Redundanz entsteht durch die Art und Weise, wie Übersetzer Code generieren (z.B. durch Schablonen). Die bisherigen PA-Ansätze betrachten das Programm als Folge von Instruktionen und suchen nach exakt gleichen Teilfolgen. Sind allerdings Instruktionen innerhalb einer Teilfolge vertauscht, wird sie nicht als weitere Instanz der gesuchten Folge erkannt. Somit fallen potentielle Code-Fragmente für die PA aus der Analyse heraus und das Ergebnis wird suboptimal. Der am Lehrstuhl untersuchte Ansatz löst dieses Problem indem die Instruktionen eines Grundblocks statt in einer Folge in einen Datenflussgraphen (DFG) umgesetzt werden. Ein Graph-Mining-Werkzeug sucht in den DFGs nach gemeinsamen Fragmenten in ARM Assembler-Code, der auf eingebetteten Systemen weit verbreitet ist. In Kooperation mit dem Projekt ParSeMiS, das sich mit der Optimierung von Graph-Minern befasst, werden auch die für

PA spezifischen Probleme beim Graph-Minen angegangen. Im Berichtszeitraum wurden die Analysen zur korrekten Rekonstruktion des Datenflussgraphen verfeinert. Eine möglichst genaue Rekonstruktion erhöht das Einsparungspotential im Vergleich zu den herkömmlichen sequentiellen Verfahren. Des Weiteren wurden verschiedene Auslagerungsmethoden optimiert. Diese dienen dazu, die von ParSeMiS als häufig eingestuften Code-Fragmente herauszuziehen und damit das Programm zu verkleinern. Die optimierten Auslagerungsmethoden zeichnen sich vor allem dadurch aus, dass sie möglichst kosteneffizient semantisch gleiche Fragmente vereinheitlichen.

2.3 ParSeMiS – die Parallele und Sequenzielle Mining Suite

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Marc Wörlein
Dr.-Ing. Alexander Dreweke
Dipl.-Inf. Tobias Werth
Laufzeit: 1.5.2006–31.12.2010
Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49 9131 85-27625
Fax: +49 9131 85-28809
E-Mail: philippsen@informatik.uni-erlangen.de

Die Arbeitsgruppe **ParSeMiS** (**Par**allele und **Se**quenzielle Graph **Mi**ning **S**uite) beschäftigt sich mit der Suche nach häufigen interessanten Strukturen in Graphdatenbanken; ein Forschungsgebiet, das in den letzten Jahren sehr reges Interesse findet. Da viele Forschungs- und Wirtschaftsdaten in strukturierter Form erfasst werden können, bietet sich die Speicherung komplexer Zusammenhänge in Form von allgemeinen oder speziellen Graphen an.

Diese meist unüberschaubaren Datenmengen sind nur schwer mit Hand und Auge zu erfassen, so dass Algorithmen zur Entdeckung interessanter Korrelationen unabdingbar sind. Da deren Entdeckung in Graphen im Allgemeinen aufwändig ist (NP-vollständig), ist die Suche nach parallelen und spezialisierten Algorithmen und Heuristiken notwendig, die den benötigen Rechenzeit- und Speicheranforderungen auch bei immer größer werdenden Datenmengen gewachsen sind.

Das Ziel dieses Projektes ist es, ein effizientes und flexibles Werkzeug zur Suche in beliebigen Graphdaten bereitzustellen, um sowohl die Einbindung in neue Anwendungsgebiete als auch die Entwicklung neuer Suchverfahren zu beschleunigen und zu verein-

fachen.

Im Jahr 2010 wurden die verteilten Stack-Implementierungen auch für andere Algorithmen und Daten getestet.

2.4 JavaParty

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Marc Wörlein
Beginn: 1.4.2007
Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49 9131 85-27625
Fax: +49 9131 85-28809
E-Mail: philippsen@informatik.uni-erlangen.de

[JavaParty]http://svn.ipd.uni-karlsruhe.de/trac/javaparty/wiki/JavaParty erlaubt eine einfache Portierung von parallelen Java-Programmen mit mehreren Threads auf eine verteilte Umgebung wie z.B. einem Cluster. Das Standard-Java unterstützt parallele Programme durch Threads und Synchronisationsmechanismen. Während Mehrprozess-Java-Programme auf einen einzelnen Speicheraddressbereich beschränkt sind, dehnt JavaParty die Möglichkeiten von Java auf verteilte Systeme aus.

Die übliche Art, parallele Anwendungen auf eine verteilte Umgebung zu portieren, ist die Verwendung von Kommunikationsbibliotheken. Java's entfernter Methodenaufruf (RMI) macht die Verwendung expliziter Kommunikationsprotokolle unnötig, führt aber immer noch zu einer erhöhten Programmkomplexität. Gründe dafür sind die beschränkten Möglichkeiten des RMIs und die benötigte zusätzliche Funktionalität für die Erzeugung und den Zugriff auf entfernte Objekte.

Der Ansatz von JavaParty ist anders. JavaParty-Klassen können direkt als entfernt (remote) deklariert werden. Während normale Java Klassen auf eine einzelne Virtuelle Maschine beschränkt sind, sind entfernte Klassen und deren Instanzen in der gesamten verteilten JavaParty-Umgebung sichtbar und erreichbar. Soweit man nur entfernte Klassen betrachtet, kann die JavaParty-Umgebung als eine Virtuelle Maschine angesehen werden, die sich über verschiedene Computer verteilt. Der Zugriff und die Erzeugung von entfernten Klassen sind syntaktisch nicht von dem regulärer Java-Klassen zu unterscheiden.

Im Jahr 2010 wurden folgende Ziele erreicht:

- Viele in der Vergangenheit selbstimplementierte Datenstrukturen des Laufzeit Systems wurden aus Stabilitätsgründen durch inzwischen effizientere Java-eigene Implementierungen ersetzt.
- Die Kommunikationsschicht (KaRMI) wurde aufgrund von Kompatibilitäts- und Sicherheitsproblemen bassierend auf der aktuellesten Java-Socket-Technologie neu implementiert.
- Entfernte Klassen wurden um einen ”nahen” Kontext erweitert, der für jedes Objekt auf jedem Rechner lokalen Speicher zur Verfügung stellt. Damit lassen sich lokalitätsgewahre Algorithmen ausdrücken.

2.5 Jackal – Cluster and Grid computing made easy

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
PD Ronald Veldema, Ph.D.
Beginn: 1.1.2006

Im Jackal Projekt wird ein Distributed Shared Memory Systems (DSM) für Java entwickelt. Das System erlaubt es, ein Programm mit mehreren Threads auf einem Rechnerbündel auszuführen. Gleichzeitig bleibt das Programm auf normalen JVMs (die nur einen Rechner unterstützen) lauffähig. Zur besseren Fehlertoleranz beinhaltet Jackal ein Checkpoint System, das in periodischen Abständen den Programmzustand auf die Festplatte speichern kann. Es ist dann möglich, diesen gespeicherten Zustand zu einem späteren Zeitpunkt zu laden und die Ausführung fortzusetzen. Außer normalen nebenläufigen Programmen lassen sich mit Jackal Anwendungen, die OpenMP Direktiven beinhalten, verwenden. Durch eine Kombination aus dem Checkpoint System und den OpenMP Direktiven kann man Anwendungen (ab dem gespeicherten Checkpoint) mit einer beliebigen Rechnerzahl wiederaufnehmen. Man ist dadurch nicht mehr an die Anzahl der Rechner gebunden, die beim Starten des Programms festgelegt wurde. Das Jackal-DSM ist nicht nur für klassische Rechnerbündel ausgelegt. Es gibt Erweiterungsarbeiten sowohl für den Einsatz im Grid als auch für Rechnerbündel mit Multicore-Prozessoren.

2.6 Tapir

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
PD Ronald Veldema, Ph.D.
Dr.-Ing. Michael Klemm
Laufzeit: 1.1.2006–31.12.2010
Kontakt:
PD Ronald Veldema, Ph.D.
Tel.: +49 9131 85-27622
Fax: +49 9131 85-28809
E-Mail: veldema@informatik.uni-erlangen.de

Tapir ist eine neue Programmiersprache zur Vereinfachung der Systemprogrammierung.

Systemprogrammierung bezeichnet unter anderem die Programmierung von Netzwerkprotokollen, Betriebssystemen, Middleware- und DSM-Systemen. Diese Komponenten stellen essentielle Teile eines Systems dar, da sie Dienste bereitstellen, auf denen andere Applikationen aufbauen. Das Betriebssystem stellt einer Anwendung z.B. eine Ausführungsumgebung bereit, die von der konkreten Hardware abstrahiert, so dass die Applikation eine robuste, Hardware-unabhängige Schnittstelle nutzen kann. Ein weiteres Beispiel für Systemprogrammierung ist ein DSM-System. Es simuliert in einem Rechnerbündel mit verteiltem Speicher einen gemeinsamen Adressraum und verbirgt die Kommunikation im Rechnerbündel vor der Anwendung.

Im Vergleich zu Anwendungssoftware stellt systemnahe Software völlig andere Anforderungen an die Programmierung. Die Leistung der einzelnen Systemkomponenten kann sich auf alle Anwendungen und somit das gesamte System auswirken. Deshalb muss der erzeugte Maschinen-Code besonders effizient ausführbar sein. Fehler auf der Systemebene beeinflussen nicht nur einzelne Anwendungen sondern können ebenfalls das gesamte System beeinträchtigen. Systemsoftware sollte aus diesem Grund möglichst (beweisbar) fehlerfrei sein. All diese Anforderungen haben direkte Auswirkungen auf die verwendbaren Programmiersprachen und den verwendeten Programmierstil:

- Hochsprachen wie C++, C# und Java verstecken Implementierungsdetails vor dem Programmierer. Der Programmierer benötigt z.B. kein Wissen darüber, wie ein Methodenaufruf konkret durchgeführt wird. Dieses Wissen ist jedoch bei der Entwicklung von Systemsoftware erforderlich.

- Hochsprachen stellen Funktionen bereit, die für Systemsoftware in der Regel nicht benötigt werden oder sogar unerwünscht sind. Beispielsweise wird innerhalb eines Betriebssystems keine automatische Speicherbereinigung oder Ausnahmebehandlung verwendet.
- Systemprogramme erfordern kein so hohes Abstraktionsniveau, wie es meist von Hochsprachen gefordert wird. Ebenso verzichtet man bei der Erstellung von Systemsoftware zumeist auf die Benutzung externer Bibliotheken, da viele Bibliotheken auf die Systemsoftware als Basis angewiesen sind und somit nicht zur Verfügung stehen.

Tapir ist an existierende Hochsprachen wie C++ und Java angelehnt, verzichtet aber auf Eigenschaften und Funktionen die ohne Bedeutung für eine Sprache zur Systemprogrammierung sind. Beispielsweise besitzt Tapir keine Speicherbereinigung, Ausnahmebehandlung oder Typumwandlung. Klassen und Objekte können zwar definiert werden, eine Vererbungsbeziehung zwischen Klassen ist aber nicht erlaubt. Das mit Tapir spezifizierte Systemprogramm kann mit Model Checking-Techniken bereits während der Entwicklung auf Fehler überprüft werden. Ein Übersetzerprototyp und ein Werkzeug zum Model Checking sind bereits implementiert. Tapir-Code ist leicht verifizierbar, da Programmteile als Implementierungsdetails markiert werden können, die danach bei der Verifizierung ignoriert werden. Tapir-Programme können (z.B. auf Grafikkarten) parallel ausgeführt werden, ohne dass es zu Verklemmungen oder ähnlichen Fehlern kommt, die üblicherweise bei paralleler Programmierung auftreten.

Schon während sich Tapir noch in der Entwicklung befindet, wurde es bereits verwendet, um eine Spezifikation für das DSM-Protokoll von Jackal zu erarbeiten. Zur Erweiterung des Tapir-Sprachentwurfes wurden RDMA-basierte DSM-Protokolle evaluiert. Die semantische Analyse von Tapir-Programmen ist sehr speicherintensiv, da sie auf Model Checking beruht. Um die Analyse zu beschleunigen, war es erforderlich, eine eigene, auf sehr große Objektmengen spezialisierte, Virtuelle Maschine für Java zu entwickeln. Die neu entwickelte, LVM genannte virtuelle Maschine zeigt wesentlich bessere Laufzeiteigenschaften als übliche Java Virtual Machine Implementierungen, sobald der verfügbare Hauptspeicher nicht mehr ausreicht und das Auslagern auf den Hintergrundspeicher notwendig wird.

2.7 OpenMP/Java

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
PD Ronald Veldema, Ph.D.

Dr.-Ing. Michael Klemm
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Laufzeit: 1.10.2009–1.10.2015

JaMP ist eine Implementierung des bekannten OpenMP Standard 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 die Funktionalität von OpenMP 2.0 und unterstützt bereits Elemente der Spezifikation 3.0 (z.B. die collapse clause). JaMP erzeugt reinen Java Code und ist damit auf jeder JVM (die Java 1.5+ unterstützt) lauffähig. Die neueste Version kann sogar CUDA fähige Hardware zur Schleifenausführung nutzen, falls der Schleifenrumpf eine Transformation nach CUDA möglich macht. Ist die Transformation nicht möglich (zum Beispiel weil ungenaue float-Divisionen verwendet werden), wird nebenläufiger Code für gängige Multicore Prozessoren erzeugt.

Im Berichtszeitraum 2010 wurde die JaMP-Umgebung erweitert, so dass die gleichzeitige Nutzung von mehreren Maschinen und Acceleratoren unterstützt wird. Dieses wurde durch die Entwicklung von zwei Abstraktionsbibliotheken erreicht. Die untere Schicht bietet abstrakte Recheneinheiten, die von den eigentlichen Berechnungseinheiten wie CPUs und GPUs und ihrem Ort in einem Rechnerbündel abstrahieren. Die darauf aufbauende Schicht bietet partitionierte und replizierte Arrays. Ein partitioniertes Array wird dabei automatisch über die abstrakten Berechnungseinheiten verteilt, wobei die Geschwindigkeiten der einzelnen Berechnungseinheiten zwecks fairer Verteilung berücksichtigt werden. Welcher abstrakte Array-Typ für ein Array eines Java-Programms konkret eingesetzt wird, entscheidet der JaMP-Übersetzer mit Hilfe einer Code-Analyse.

2.8 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
Kontakt:
Dipl.-Inf. Stefan Kempf
Tel.: +49-9131-85-27624
Fax: +49-9131-85-28809
E-Mail: stefan.kempf@informatik.uni-erlangen.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.

Dazu bearbeitet dieses Projekt drei Bereiche. Der erste Bereich 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 kombiniert 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.

Dieser Übersetzer wird auch für den zweiten Aspekt des Projekts, der Parallelisierung von Programmen verwendet. Zunächst wurden Programme aus der Automatisierungstechnik auf ihre automatische Parallelisierbarkeit untersucht. Hierbei hat sich herausgestellt, dass diese wie bei der Migration 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 bereits im Berichtszeitraum 2009 begonnene Ausführungsumgebung für parallelisierte Programme wurde weiterentwickelt. Diese Bibliothek 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.

Mit der Entwicklung eines lernfähigen Expertensystems zur Unterstützung von Entwicklern in der Optimierung und Restrukturierung von Programmen wurde im Berichtszeitraum 2010 begonnen. 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 Quell-Code, 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 Quell-Code können diese Muster für die Datenbank generiert werden.

2.9 Modellgetriebene Komponentenkomposition

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Philipp Janda
Laufzeit: 15.6.2007–14.6.2011
Förderer: AUDI AG

Dieses im Rahmen der INI.FAU-Kooperation durchgeführte Projekt analysiert die Integration von Fahrzeugfunktionen auf Steuergeräte und entwickelt modellgetriebene Unterstützungsmöglichkeiten. Die gewonnenen Erkenntnisse werden exemplarisch anhand der Integration aller Komponenten eines Fahrdynamikregelsystems auf einem AUTOSAR-Steuergerät überprüft werden.

In der Automobilindustrie ist es schon lange üblich, Fahrzeugfunktionen auf hohem Abstraktionsniveau modellbasiert zu entwickeln. Um frühzeitig Fehleinschätzungen bezüglich Laufzeit- und Ressourcenbedarf auszuschließen, ist es nötig, die entwickelte Software nicht nur zu simulieren, sondern auch auf der Zielhardware testen zu können. Aufgrund von Kosten- und Sicherheitsanforderungen ist die Integration auf ein Steuergerät aber sehr zeitaufwändig und erfordert Expertenwissen, das einem Funktionsentwickler normalerweise nicht zur Verfügung steht. AUTOSAR (AUTomotive Open System ARchitecture) etabliert sich als Standard für die Basissoftware auf Steuergeräten. Doch durch die Neuheit dieses Standards gibt es noch keine Verfahren und Werkzeuge, um die Integration von Funktionen auf einem Steuergerät zu unterstützen.

Im Jahr 2008 wurden die Modellierungsmöglichkeiten in AUTOSAR in Bezug auf ihre Eignung bei Audi und auf mögliche Konflikte mit bestehenden Standards sowie mit bei Audi eingesetzten Technologien untersucht. Des Weiteren wurde die automatische Ver-

vollständigung einer Reglerkomponente zu einer AUTOSAR-Softwarearchitektur prototypisch realisiert.

Im Jahr 2009 wurde damit begonnen, die ermittelten Unterstützungsmöglichkeiten (automatische Konfiguration der Buskommunikation mit Hilfe einer Busdatenbank sowie automatisches Scheduling der auszuführenden Prozesse) mit Hilfe eines modellgetriebenen Ansatzes auf Basis des Eclipse Modeling Frameworks zu realisieren und in die Werkzeuglandschaft bei Audi zu integrieren. Durch die modellgetriebene Entwicklung wird eine leichte Anpassbarkeit des entstehenden Prototypen an sich verändernde Anforderungen erzielt.

Im Jahr 2010 wurde die in einem AUTOSAR-Projekt zur Verfügung stehenden Informationen genutzt, um daraus automatisch sowohl die lokale Kommunikation zu konfigurieren, als auch diejenige, die Steuergerätegrenzen überschreitet. Ebenso konnten bereits vorhandene Abhängigkeitsspezifikationen mit Hilfe eines genetischen Algorithmus zur automatischen Erzeugung eines Taskscheduling verwendet werden, der die Kommunikationslatenzen zwischen kooperierenden Softwarebausteinen minimiert. Der bestehende Prototyp wurde um die erarbeiteten Verfahren erweitert.

2.10 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
Al-Hilank, Samir
Laufzeit: 1.10.2008–30.9.2011
Förderer: Bundesministerium für Wissenschaft und Technologie

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 soll im Rahmen eines laufenden Kooperationsprojektes geschlossen werden, das in Zusammenarbeit mit der develop group als Industriepartner durchgeführt und mit Mitteln des BMWi gefördert wird. Es wurde im Oktober 2008 mit drei wissenschaftlichen Mitarbeitern gestartet und ist auf insgesamt drei Jahre ausgelegt.

Ziel dieses Kooperationsprojektes ist 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 liegt 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 sollen 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 werden.

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. Eine Zusammenfassung der im eSPEM definierten Erweiterungen des SPEM wurde anlässlich der ”6th European Conference of Modelling Foundations and Applications” veröffentlicht.

Der im Rahmen des Projektes entwickelte Prototyp einer eSPEM-basierten Modellierungsumgebung für SWEPe wurde 2010 maßgeblich erweitert. Mit Hilfe dieses Prototypen konnte das eSPEM durch die exemplarische Modellierung von industrierelevanten SWEPen validiert und verbessert werden. Neu erstellt wurde eine prototypische Ausführungsmaschine, die eSPEM-basierte Prozessmodelle instanzieren, simulieren und verteilt ausführen kann. Diese Ausführungsmaschine wird derzeit anhand von Testmodellen validiert.

2.11 Softwareleitstand – Prototypische Entwicklung eines neuartigen Werkzeugs zur Qualitätsabsicherung bei der Softwareentwicklung

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. (FH) Josef Adersberger
M. Eng. Norbert Tausch, Dipl.-Ing. (FH)
Beginn: 1.11.2009
Förderer: Bundesministerium für Wirtschaft und Technologie
Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49 9131 85-27625
Fax: +49 9131 85-28809
E-Mail: philippsen@informatik.uni-erlangen.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 aus Mitteln des BMWi gefördert. Die Projektlaufzeit ist November 2009 bis Mai 2012.

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. Im Berichtszeitraum wurde das Konzept für die Umsetzung des Analysekerns erstellt und dessen Implementierung begonnen. Der Analysekern ist für die Untersuchung der als Objektgraph vorliegenden Traceability-Daten verantwortlich und greift auf graphentheoretische Aspekte zurück. Die Formulierung von Analysen und zu prüfenden Regeln soll dabei mittels einer geeigneten Abfragesprache möglich sein. Hierzu wurde aufbauend auf dem Konzept des Analysekerns eine Studie bzgl. der Verwendbarkeit bereits existierender Sprachen vorgenommen. Im Ergebnis wurde die objekt-orientierte und funktionale Multiparadigmensprache Scala als Grundlage für die Umsetzung der Abfragesprache in Form einer Domain Specific Language beschlossen. Die DSL ist bzgl. der Realisierung in ihrer ersten Ausbaustufe funktionsfähig und ermöglicht beispielsweise das Erstellen von sog. Reflexion Modellen. Diese bilden Übereinstimmungen und Differenzen zwischen einem Komponenten- und einem Code-Modell mittels neuer Beziehungen im Komponentenmodell ab. Weiterhin ist das Verdichten von Metriken aus dem Code-Modell in das Komponentenmodell möglich.

2.12 Funkortung

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Ing. (FH) Thorsten Edelhäußer
Dipl.-Inf. Christopher Mutschler
Laufzeit: 1.5.2008–14.11.2013
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 vielen weiteren Prozessen. 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 Funkortung die automatisierte Einmessung von Funkortungssystemen, die Generierung von generischen dynamischen Bewegungsmodellen und Verfahren zur Ereignisdetektion in Positionsdatenströmen zur Laufzeit.

Im Jahr 2009 wurden Algorithmen zur Berechnung der Position und Orientierung der Empfangsantennen eines Funkortungssystems weiterentwickelt. Die Algorithmen berechnen selbstständig Messpunktkoordinaten, die eine schnelle und genaue Einmessung ermöglichen. Zur automatisierten Einmessung wurden hierzu Roboter verwendet. Unser Algorithmus berücksichtigt Hindernisse und die Empfangseigenschaften des Ortungssystems und kann Fehlmessungen wie Mehrwegeempfang in der Berechnung der Position und Orientierung der Empfangsantenne aussortieren.

2010 wurden Modelle entwickelt, die dynamische Bewegnungsmodelle ermöglichen. Hier wurden lernende Verfahren eingesetzt, um Modelle während der Laufzeit anzupassen. Es wurde die formale Sprache TBL (Trajectory Behavior Language) konstruiert, mittels derer Trajektorien beschrieben werden können. Weitere Algorithmen können die Darstellung nochmals verkleinern und somit die zur Speicherung von Trajektorien erforderliche Datenmenge komprimieren.

Nun werden Verfahren untersucht, mit denen die entwickelten Bewegungsmodelle zur Laufzeit gelernt werden können. Diese sollen in einer Untersuchung zur Prädiktion von Bewegungen evaluiert werden. Desweiteren wird untersucht, inwieweit sich auftretende Ereignisse in Lokalisierungssystemen vorhersagen lassen können, indem vorhandene

Ereignisströme zur Laufzeit analysiert und gelernt werden.

2.13 Graphen und Graphtransformationen

Projektleitung:
Prof. em. Dr. Hans Jürgen Schneider
Laufzeit: 1.10.2004–30.9.2012
Kontakt:
Prof. em. Dr. Hans Jürgen Schneider
Tel.: +49 9131 85-27620
Fax: +49 9131 85-28809
E-Mail: schneider@informatik.uni-erlangen.de

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

Die zugrunde liegende Theorie 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.

Auch im Jahre 2010 haben wir sowohl theoretische Konzepte untersucht als auch einen Implementierungsaspekt betrachtet:

- Im Mittelpunkt unserer Arbeiten über Graphtransformationen steht der sogenannte Doppelpushout-Ansatz, bei dem eine Produktion durch zwei Morphismen bestimmt wird, die an einem gemeinsamen Interface-Graphen ansetzen. Die Weiterarbeit an dem geplanten Lehrbuch hat dazu geführt, einige Alternativen zu analysieren und deren Beziehungen zum Doppelpushout-Ansatz zu studieren. Wir haben zunächst die Hyperkantenersetzung und die einwärts gerichteten Produktionen betrachtet. Die Hyperkantenersetzung dient der formalen Beschreibung sowohl von Berechnungen als auch von graphischen Darstellungen. In der Literatur

wird sie üblicherweise nicht im Kontext des Doppelpushout-Ansatzes behandelt, weil die meisten Arbeiten über diesen nichtinjektive Abbildungen ausschließen. Unsere bisherigen Ergebnisse vermeiden diese Einschränkung, und deshalb passt die Hyperkantenersetzung zu dem Doppelpushout-Ansatz. Als Nebeneffekt finden wir eine Klasse kontextfreier Graphgrammatiken, die einige Sprachen umfassen, die als Zeichenkettengrammatiken nicht kontextfrei sind. Das zweite Thema basiert auf einem Vorschlag von Banach. Er hat die Richtung der Morphismen in der Produktion umgedreht, so dass sie nach Innen gehen und die linke und die rechte Seite in einen größeren Kontext einbetten. Dieser Ansatz erlaubt die Aufdeckung einiger interessanter Beziehungen zu operationellen (nicht-kategoriellen) Ansätzen, aber auch zu dem Ansatz der "adhesive grammar". Die Details werden in Kürze im Entwurf des Lehrbuches veröffentlicht: http://www2.informatik.uni-erlangen.de/Personen/schneide/gtbook/chapter6.pdf

- Die kategorielle Behandlung der Graphtransformationen ist hochgradig generisch: Alle Beweise und Konstruktionen gelten für verschiedene Graphtypen. Nur die grundlegenden Operationen müssen für jede Anwendung detailliert beschrieben werden, die darauf aufbauenden kategoriellen Eigenschaften sind dann automatisch definiert. Da moderne Programmiersprachen generische Konzepte unterstützen, sieht es vielversprechend aus, den kategoriellen Ansatz zur Beschreibung von Graphtransformationssystemen in Sprachen wie Java oder Haskell zu implementieren. Im Jahre 2008 haben wir diesen Ansatz unter Ausnutzung des Polymorphismus in der funktionalen Sprache Haskell implementiert. Java benutzt Klassen von Objekten, unterstützt aber nicht wirklich die mehrfache Vererbung. In der Version von 2009 definieren wir Schnittstellen Cat, CatWithColimits usw, die durch Klassen wie beispielsweise CatFinSet oder CatFinGraph implementiert werden müssen. Diese Lösung erlaubt jedoch nicht, die Methode zur Konstruktion von Pushouts in CatWithColimits aufzunehmen. Deshalb haben wir eine separate "Factory"-Klasse PushoutCreator definiert, die die kategorielle Konstruktion unabhängig von einer speziellen Kategorie beschreibt und von den die speziellen Kategorien beschreibenden Klassen importiert werden muss. Jetzt haben wir eine Alternative implementiert, die auf den generischen Datentypen von Java basiert. Der Trick besteht darin, eine abstrakte Klasse CocoCat der covollständigen Kategorien zu definieren, die die kanonischen Konstruktionen der Colimites, z.B. des Pushouts, enthält und sie so den konkreten Unterklassen zur Verfügung stellt. Der entscheidende Nachteil dieser Vorgehensweise ist, dass die generischen Parameter bei allen Klassen und allen Methoden mitgeschleppt werden müssen. Eine Zusammenfassung dieser Implementierung ist zu finden in Engels et al. (Eds.): Graph Transformation and Model-Driven Engineering (Lect. Notes Comp. Sc. vol. 5765, 2010), pp. 33-58.

2.14 International Collegiate Programming Contest an der FAU

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Marc Wörlein
Dr.-Ing. Alexander Dreweke
Dipl.-Inf. Daniel Brinkers
Beginn: 1.11.2002
Kontakt:
Dipl.-Inf. Tobias Werth
Tel.: +49 9131 85-28865
Fax: +49 9131 85-28809
E-Mail: tobias.werth@informatik.uni-erlangen.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 zehn 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 der 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 (2011 in Sharm-el-Sheikh, Ägypten) stattfinden. Im Jahr 2010 fanden zwei lokale Wettbewerbe an der FAU statt. Im Wintersemester wurde ein Mannschaftswettbewerb ausgetragen mit dem Ziel, neue Studierende für die Wettbewerbe zu begeistern mit einer Rekordzahl von 24 Teams. Jedes Team bestand aus maximal drei Studenten. Außerdem nahmen noch 13 Teams der TU München sowie zwei Teams der Universität Konstanz online am Wettbewerb 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 2010.

Insgesamt nahmen an unserem erstmalig deutschlandweit organisierten Ausscheidungskampf 15 Teams der FAU mit Studenten verschiedensten Fachrichtungen teil. 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 gewann den nordwesteuropäischen Wettbewerb NWERC in Bremen souverän und setzte sich gegen die internationale Konkurrenz durch. Das zweite Team erzielte ebenfalls ein hervorragendes Ergebnis und landete auf dem 4. Platz. Der Erfolg wurde durch das dritte Team abgerundet, das im Mittelfeld der 67 Teams Platz 33 erreichte. Auch 2010 zeigte das Trainingslager somit den gewünschten Erfolg, da das beste Team zu den Weltmeisterschaften in Sharm-el-Sheikh im Frühjahr 2011 fahren und dort als einziges deutsches Team teilnehmen darf.

2.15 Embedded Systems Institute

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Philipp Janda
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
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

3 Publikationen

- Dotzler, Georg ; Veldema, Ronald ; Klemm, Michael: JCudaMP: OpenMP/Java on CUDA . In: Pankratius, Victor ; Philippsen, Michael (Hrsg.) : Proceedings of the Third International Workshop on Multicore Software Engineering (IWMSE10) (International Workshop on Multicore Software Engineering Cape Town, South Africa 01.05.2010). Los Alamitos, CA, USA : ACM Press, 2010, S. 10-17. - ISBN 978-1-60558-964-0

- Dreweke, Alexander: Graphbasierte Prozedurale Abstraktion . Erlangen, Friedrich-Alexander-Universität, Diss., 2010. - 201 Seiten.

- Edelhäußer, Thorsten ; Frühauf, Hans-Holm: Concept for encoding data defining coded orientations representing a reorientation of an object . Schutzrecht 10196868.3 DE Patentanmeldung (23.12.2010)

- Edelhäußer, Thorsten ; Frühauf, Hans-Holm ; Kókai, Gabriella ; Philippsen, Michael: Concept for encoding data defining coded positions representing a trajectory of an object . Schutzrecht 10196851.9 DE Patentanmeldung (23.12.2010)

- Edelhäußer, Thorsten ; Kókai, Gabriella ; Frühauf, Hans-Holm: Design for determining an estimated value for a position of a reception element . Schutzrecht WO/2010/128121 DE Patentschrift (11.11.2010)

- Edelhäußer, Thorsten ; Janak, Mateusz ; Kókai, Gabriella: Environment-Based Measurement Planning for Autonomous RTLS Configuration . In: IEEE (Hrsg.) : Proceedings of the Conference on Adaptive Hardware and Systems (AHS 2010) (Conference on Adaptive Hardware and Systems (AHS 2010) Anaheim California, USA). 2010, S. 289-296.

- Ellner, Ralf ; Al-Hilank, Samir ; Drexler, Johannes ; Jung, Martin ; Kips, Detlef ; Philippsen, Michael: eSPEM - A SPEM Extension for Enactable Behavior Modeling . In: Kühne, Thomas ; Selic, Bran ; Gervais, Marie-Pierre ; Terrier, Francois (Hrsg.) : 6th European Conference of Modelling Foundations and Applications (ECMFA 2010, 6th European Conference of Modelling Foundations and Applications Paris, France June 15-18, 2010). Bd. 6138. Berlin : Springer, 2010, S. 116-131. (Lecture Notes in Computer Science Bd. 6138) - ISBN 978-3-642-13594-1

– Haumacher, Bernhard ; Philippsen, Michael ; Tichy, Walter F.: Irregular data-parallelism in a parallel object-oriented language by means of Collective Replication . Erlangen : Universität Erlangen, Dept. of Computer Science 2. 2010 (CS-2010-04). - Interner Bericht. 14 Seiten (Department Informatik - Technical Reports Bd. CS-2010-04) ISSN 2191-5008

– Schneider, Hans Jürgen ; Minas, Mark: Graph transformation by computational category theory . In: Engels, Gregor ; Lewerentz, Claus ; Schäfer, Wilhelm ; Schürr, Andy ; Westfechtel, Bernhard (Bearb.): Graph Transformations and Model-Driven Engineering - Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday. Berlin/Heidelberg : Springer, 2010, (Lecture Notes Computer Science Bd. 5765), S. 33-58. - ISBN 978-3-642-17321-9

– Veldema, Ronald: Improved DSM Efficiency, Flexibility, and Correctness . Erlangen, FAU, Habil-Schr., 2010 (Technical reports / Department Informatik Bd. CS-2010,3) . - 144 Seiten. ISSN 2191-5008

– Veldema, Ronald ; Philippsen, Michael: Safe and Familiar Multi-core Programming by means of a Hybrid Functional and Imperative Language . In: Gao, Guang R. ; Pollock, Lori L. ; Cavazos, John ; Li, Xiaoming (Hrsg.) : Languages and Compilers for Parallel Computing (22nd International Workshop of Languages and Compilers for Parallel Computing (LCPC 2009) Delaware October 8-10, 2009). Bd. 5898. Berlin : Springer, 2010, S. 157-171. - ISBN 978-3-642-13373-2

– Werth, Tobias: DOMjudge - An Automated Judging System for Programming Contests .Vortrag: Collaborative Learning Institute Symposium, Collaborative Learning Institute, Harbin, China, 04.02.2010

– Zvada, Szilvia: Attribute Grammar Based Genetic Programming . Erlangen, FAU, Diss., 2010. - 213 Seiten.

4 Studien- und Abschlussarbeiten

– Studienarbeit: Automatisches und integriertes Testen von elektronischen Bauelementen. Bearbeiter: Eberlein Stefan (beendet am 20.01.2010); Betreuer: PD Ronald Veldema, Ph.D.; Dipl.-Ing. Sabine Walther

– Diplomarbeit: Definition von Entwicklungsprozessen für Kollaborationsplattformen am Beispiel von IBM Rational Jazz und Team Concert. Bearbeiter: Gabriel Dexheimer (beendet am 03.02.2010); Betreuer: Hon.-Prof. Dr.-Ing. Bernd Hindel; Prof. Dr. Michael Philippsen

– Studienarbeit: Interaktive Prozesssteuerung für ein verteiltes Zeitplanungssystem. Bearbeiter: Kerim Merdenoglu (beendet am 16.02.2010); Betreuer: PD Dr.-Ing. habil. Peter Wilke

– Diplomarbeit: Carolo-Cup: Bewegungsplanung für das Fahrzeug. Bearbeiter: Philipp Sommer (beendet am 17.2.2010); Betreuer: apl. Prof. Dr.-Ing. Gabriella Kókai; Prof. Dr. Michael Philippsen

– Studienarbeit: Case-Study: Untersuchung des industriellen Einsatzes von .gEAr im web-basierten Client-Server-Umfeld. Bearbeiter: Renato Gabriel Giurea (beendet am 01.4.2010); Betreuer: Dr.-Ing. Norbert Oster; Prof. Dr. Michael Philippsen

– Diplomarbeit: Mehrstufige Parallelisierung von Graph-Mining-Algorithmen. Bearbeiter: Bernd Schöbel (beendet am 03.05.2010); Betreuer: Dipl.-Inf. Marc Wörlein; Prof. Dr. Michael Philippsen

– Diplomarbeit: Online Data-Mining of trajectories in Real Time Location Systems. Bearbeiter: Christopher Mutschler (beendet am 12.05.2010); Betreuer: apl. Prof. Dr.-Ing. Gabriella Kókai; Prof. Dr. Michael Philippsen; Dipl.-Ing. (FH) Thorsten Edelhäußer

– Studienarbeit: Konzeptionierung und Pilotierung eines automatisierten Deploy- und Test-Systems. Bearbeiter: Fabian Zach (beendet am 01.07.2010); Betreuer: Dr.-Ing. Norbert Oster; Prof. Dr. Michael Philippsen

– Diplomarbeit: Reengineering des modellbasierten Testdatengenerators UnITeD hinsichtlich der verwendeten Heuristiken zwecks Optimierung der Performance und Ergebnisqualität. Bearbeiter: Kai Guo (beendet am 26.07.2010); Betreuer: Dr.-Ing. Norbert Oster; Prof. Dr. Michael Philippsen

– Diplomarbeit: Multi-GPU Cluster use for Java/OpenMP. Bearbeiter: Thorsten Blaß (beendet am 02.08.2010); Betreuer: PD Ronald Veldema, Ph.D.; Prof. Dr. Michael Philippsen

– Diplomarbeit: Analytical and Experimental Evaluation of SMP Superscalar. Bearbeiter: Jan Ciesko (beendet am 2.11.2010); Betreuer: PD Ronald Veldema, Ph.D.; Prof. Dr. Michael Philippsen