Lehrstuhl für Informatik 2

(Programmiersprachen und Programmiermethodik)

Leiter:	**Prof. Dr. Hans Jürgen Schneider**

Mitarbeiter:

Allendorf, Helmut, Dipl.-Ing. (FH)	Programmierer
Billing, Gunnar, Dipl.-Inf.	wiss. Mitarbeiter bis 30.09.2000
Brütting, Agnes	Sekretärin - halbtags
Dormeyer, Ricarda, Dipl.-Inf.	wiss. Mitarbeiter, beurlaubt bis 30.04.00
Fischer, Ingrid, Dr.-Ing.	wiss. Mitarbeiter, ab 29.11.00 beurlaubt
Gröbner, Matthias, Dipl.-Inf.	wiss. Mitarbeiter
Hodek, Roman, Dipl.-Inf.	wiss. Mitarbeiter bis 31.07.2000
König, Roman, Dr.-Ing.	wiss. Mitarbeiter
Minas, Mark, Dr.-Ing.	Akad. Rat
Schmidt, Harald, Dipl.-Inf.	wiss. Mitarbeiter
Schörmal, Elfriede	Sekretärin - halbtags
Uebler, Manfred	Programmierer
Wilke, Peter, PD Dr.-Ing. habil.	Akad. Oberrat, beurlaubt bis 30.11.00

Stipendiaten:

Kókai, Gabriella, Dr.-Ing.	(Bayer. Wissenschaftsministerium)
Nagorny Stanislav	(Graduiertenkolleg)
Tóth Zoltán	(DAAD - Siemens)
Ványi Róbert	(DAAD - Siemens)
Zvada Silvia	(Bayer. Wissenschaftsminiserium)

Lehrbeauftragte:

Dussa-Zieger, Klaudia, Dr.-Ing.	(Fa. imbus)
Hindel, Bernd, Dr.-Ing.	(Fa. 3SOFT)
Kips, Detlef, Dr.-Ing.	(Fa. BASYS, Erlangen)
Kreisel, Klaus, Dr.rer.nat.	(Emmy-Noether-Gymnasium, Erlangen)

Kooperationspartner:

- Arbeitsgemeinschaft Software-Qualität Franken e.V.
- Bayerischer Forschungsverbund Software-Engineering
- ESPRIT-Programme APPLIGRAPH (Applications of Graph Transformation)
- EU - TMR Network GETGRATS (General Theory of Graph Transformations)
- József Attila Universität Szeged, Ungarn (Dr. Gyimóthy)
- Technische Universität Berlin (Prof. Dr. Ehrig)
- Technische Universität München (Prof. Dr. Broy)
- Universität Antwerpen, Belgien (Prof. Dr. Janssens)
- Universität Bremen (Prof. Kreowski, Dr. Hoffmann)
- Universität Gent (Dr. Bellegham)
- Universität Kaiserslautern (Prof. Litz)
- Universität der Bundeswehr, Neubiberg (Prof. Dr. Schürr)
- Universität Karlsruhe (Prof. Rembold)
- Universität Krakau, Polen (Dr. Grabska)
- Universität Pisa, Italien (Prof. Montanari)
- Universität Stuttgart, IPVR (Prof. Levi)
- Universität Tübingen (Prof. Zell)
- University of California, Berkeley (Dr. Berthold)
- University of Calgary (Prof. Prusinkiewicz)

- University of Western Australia (Prof. Bräunl)
- Astrum GmbH, Erlangen
- Audi AG, Ingolstadt
- Basys GmbH, Erlangen
- Deutscher Wetterdienst, Offenbach/Potsdam
- FAST e.V., München
- FZI Forschungszentrum Informatik, Karlsruhe (Dr. Berns)
- GaG-GmbH, Greding-Obermässing
- imbus GmbH, Erlangen
- INA Wälzlager Schaeffler KG, Herzogenaurach
- Infoteam GmbH, Bubenreuth
- ipcas, Tennenlohe
- Nureg, Nürnberg
- Siemens AG, Abt. ATD IT PS 15 Erl
- Siemens A & D, Nürnberg
- 3SOFT GmbH, Erlangen

Forschungsergebnisse:

1 Computability in an Introductory Course on Programming

A usual programming course is mainly concerned with presenting syntactic sugar and semantic tricks to solve more or less small, standardized problems. In the end, the students know a lot of details how to make a computer run, and at the best, they are familiar with good design practice and some software-engineering tools. But, do they understand what computer science is and what it can bring about? To remedy this, they need not only be taught the theory as a supplement, but they should be supplied with the relationship between fundamentals and practice.

The main task of both programming and theory is making definitions, proving properties of the objects introduced by the definitions, and taking advantage of these properties. We can derive profit from this analogy by teaching theory from a programmer's point of view. A topic, which is very closely related to programming and at the same time, is at the heart of theoretical computer science, is computability

theory. Therefore, it is well-suited as an entrance to theory. Even freshmen perceive the clarity of Kleene's theory of recursive functions if the definitions are translated one-to-one into higher-order functions of about five lines using some LISP-dialect. The students can then apply these functions to create computable functions one after the other, make them run and can convince themselves that the theory works. It is nice to see that even hackers can be tempted into studying the theory of recursive functions, efficiency problems, denotational semantics, etc., if they can do this sitting in front of their beloved computers.

The programming approach to computability presented in the textbook by Kfoury, Moll, and Arbib in 1982 is a good basis to integrate theory and practice. It has been embedded into a programming course following the textbook by Abelson and Sussman. This leads to a course concept teaching good programming practice and clear theoretical concepts simultaneously. Here, we explain some of the main points of this approach: the halting problem, primitive and µ-recursive functions and the operational counterparts of these functions, i.e., the LOOP- and the WHILE-programs.

To appear in: Bulletin of the European Association for Theoretical Computer Science (EATCS) vol. 73 (2001)

See: http://www2.informatik.uni-erlangen.de/IMMD II/Persons/schneide/publications/comput.html

(Prof. Dr. H. J. Schneider)

2 Requirements Engineering in der Automatisierungstechnik

Bei der Entwicklung von Softwareprodukten kommt der Phase des Requirements Enginneering (RE) eine entscheidende Bedeutung zu. Werden fehlende, widersprüchliche oder mehrdeutige Anforderungen in dieser Phase nicht entdeckt, so führt dies i.d.R. zu einem deutlich erhöhten Änderungsaufwand in den folgenden Entwicklungsphasen sowie im ungünstigsten Fall zur Entwicklung des falschen Produkts. Eine hohe Qualität der Anforderungsspezifikation kann vielfach nur durch spezifisch auf die Anwendungsdomäne eines Softwareprodukts abgestimmte Vorgehensweisen und Verfahren erreicht werden. Im Teilprojekt A4 des Forschungsverbunds Software Engineering FORSOFT [1] wurde diese Problemstellung für zwei sich ergänzende Teilbereiche der Automatisierungstechnik aufgegriffen: die Entwicklung komplexer Standardsoftware für Automatisierungssysteme und die Entwicklung von Automatisierungssoftware für Produktionsanlagen.

Im Anwendungsbereich komplexer Standardsoftware wurde ein Vorgehens- und Datenmodell für das RE entwickelt, das sich auf bewährte Vorgehensweisen wie das Concurrent Development stützt. Das Vorgehensmodell ermöglicht es, Anforderungen ausgehend von einer Anforderungsdatenbank aus Marketingsicht in eine technische Systemsicht umzustrukturieren und diese schließlich in konkrete Anforderungsspezifikationen für einzelne Versionen eines Standardsoftwareprodukts zu überführen. Ein

Schwerpunkt war in diesem Zusammenhang die Entwicklung eines Verfahrens für die Kosten/Nutzen-Priorisierung von Anforderungen auf Grundlage der Portfoliotechnik.

Im anderen Anwendungsbereich, der Entwicklung von Automatisierungssoftware, wurde für Produktionsanlage ein Vorgehens- und Datenmodell für die integrierte Hardware- und Softwareplanung entwickelt. Die Methodik unterstützt die systematische Ermittlung und Spezifikation der Anforderungen an die Teilsysteme des Automatisierungssystems parallel zur Hardwareplanung, wobei als Ergebnis eine integrierte Designspezifikation der Mechanik-, Elektr(on)ik- und Softwareteilsysteme vorliegt. Einen Schwerpunkt bildete dabei die Integration von STEP/EXPRESS und UML. Die Ergebnisse in beiden Anwendungsbereichen wurden in enger Kooperation mit den Industriepartnern Siemens A & D AS und Homag AG sowie dem Institut für Werkzeugmaschinen und Betriebswissenschaften der TU München (iwb) erarbeitet und stehen in Form eines Abschlußberichts zur Verfügung [2].

[1] Forschungsverbund Software Engineering, http://www.forsoft.de

[2] Abschlußbericht des Forschungsverbunds Software Engineering,

http://www.forsoft.de/publikationen/bericht99.ps

(Billing)

3 Das Verhältnis natürlicher Sprache und OCL

Die Object Constraint Language (OCL) ist eine formale Sprache zur Beschreibung von Constraints, die besonders in Kombination mit der Unified Modelling Language (UML) eingesetzt wird. In Lehrbüchern zur UML herrscht allerdings Uneinigkeit, ob Constraints innerhalb vom UML-Diagrammen mit Hilfe natürlicher Sprache oder OCL beschrieben werden sollen. Im Gegensatz zu OCL-Formeln ist natürliche Sprache einfach zu verstehen, aber mehrdeutig.

Es wurde die Frage behandelt, wie OCL in natürliche Sprache und umgekehrt übersetzt werden kann, um die Vorteile beider Verfahren zu verbinden. Dies soll allerdings nicht ohne Rückgriff auf UML-Diagramme geschehen, da sie den Kontext der Constraints, sei er in natürlicher Sprache oder in OCL geschrieben, darstellen.

Basierend auf einem OCL-Parser, einem Diagrammeditor für UML-Klassendiagramme und einem auf Dependenzen und Merkmalsstrukturen basierenden Parser für natürliche Sprache wird aktuell eine Versuchsumgebung entwickelt, um die Übertragbarkeit zwischen OCL-Constraints und Constraints in natürlicher Sprache zu untersuchen.

(Fischer)

3.1 Mädchen + Technik Praktikum 2000

Zum zweiten Mal wurde das "Mädchen + Technik Praktikum 2000" an der Technischen Fakultät der Universität Erlangen Nürnberg und dem Fraunhofer Institut für Integrierte Schaltungen IIS A veranstaltet. Der Lehrstuhl für Programmiersprachen und Programmiermethodik beteiligte sich auch dieses Mal an der Organisation. Drei Mitarbeiterinnen halfen bei der problemlosen Abwicklung des Praktikums von der Anwerbung von Praktikumsversuchen, über die Ausschreibung des Praktikums an den Schulen bis zur Abwicklung der eigentlichen Praktikumswoche.

In diesem Rahmen wurde von Dr. G. Kókai ein Praktikum "Wenn Darwin programmieren würde: Evolutionäre Algorithmen - Optimierung per Evolution" angeboten, an dem drei bis vier Mädchen pro Tag die Grundlagen der Evolutionären Programmierung an Hand von Beispielen erproben konnten.

(Fischer, Kokai, Dormeyer)

Entwicklung des genetisch logischen Programmiersystems GeLog

Das Ziel des GeLog-Systems ist, die Vorteile der Methoden der genetischen Algorithmen und induktiv logischen Programmierung zu nutzen. Dieses System ist auch bei umfangreichen Problemstellungen noch effizient einsetzbar und lässt sich sehr leicht auf das Erlernen von Zusammenhängen aus verschiedenen Aufgabenbereichen anpassen. Mit dem automatischen Lernsystem Gelog können logische Programme erzeugt werden, die eine Lösung für eine gegebene Aufgabe darstellen. Die erlernten Programme liegen anschließend als Quelltext in der logischen Programmiersprache PROLOG vor und sind in dieser Form direkt ausführbar.

Zur Formulierung der Aufgabenstellung werden aus der induktiv logischen Programmierung bekannte Komponenten verwendet. Anhand von Beispieldaten erlernt das System die logischen Zusammenhänge und formuliert diese mit Hilfe von Bausteinen aus dem Hintergrundwissen als PROLOG-Programm. Der eigentliche Lernvorgang basiert auf einem genetischen Algorithmus. Zu Beginn wird aus Elementen des Hintergrundwissens zufällig eine Menge von Programmen erzeugt, die eine sogenannte Population von Individuen bildet. Diese Individuen durchlaufen einen wiederkehrenden Evolutionszyklus, in dem Generation für Generation das Erbgut der Individuen weitergegeben, kombiniert und mutiert wird. Dabei wird der Lernfortschritt und die Eignung jedes einzelnen Individuums mit Hilfe der Beispieldaten ermittelt. Daraus leitet sich die Wahrscheinlichkeit ab, mit der die Erbinformation eines Individuums in die nächste Generation weitergegeben wird.

Hinsichtlich der Kombination von genetischem Algorithmus mit induktiv logischer Programmierung wurden verschiedene Ansätze verfolgt. Um beispielsweise ein Programm an das Beispielwissen anzupassen, wurden die Operatoren zur Rekombination und Mutation so implementiert, dass sie die

gleichen Auswirkungen auf die Erbinformation wie die Generalisierungs- und Spezialisierungsmethoden der induktiv logischen Programmierung haben. Genetische Algorithmen sind in der Lage, auch große Ergebnisräume auf effiziente Weise zu durchsuchen und die darin enthaltenen vielversprechenden Regionen ausfindig zu machen. Dies ist ein entscheidender Vorteil für die Bearbeitung umfangreicher und komplexer Problemstellungen. Die aus der induktiv logischen Programmierung stammenden Elemente Hintergrundwissen und Beispieldaten erlauben dabei eine einfache und flexible Anpassung des genetischen Algorithmus an die jeweilige Aufgabenstellung, ohne dass Eingriffe in das System selbst nötig sind. Durch eine universelle Datenrepräsentation und die vielseitigen Möglichkeiten den Evolutionsverlauf durch Parameter zu beeinflussen, kann mit Hilfe des GeLog-Systems eine Vielzahl verschiedener Problemstellungen bearbeitet werden.

(Kókai)

5 Didaktik der Informatik

Die Didaktik der Informatik befindet sich, wie die Informatik als Schulfach in Bayern, im Umbruch, sowohl vom Organisatorischen als auch von den Inhalten.

An den bayerischen Gymnasien wird die Informatik nach der nächsten Änderung der Stundentafeln in etwa zwei Jahren ein eigenständiges Schulfach; alle Zweige des Gymnasiums haben dann in der 6. Jahrgangsstufe zwei Wochenstunden Informatik; in den mathematisch-naturwissenschaftlichen Gymnasien wird sie zusätzlich mit je zwei Wochenstunden in den Jahrgangsstufen 9 bis 11 unterrichtet. Es wird nicht ausbleiben, dass die Informatik schließlich auch als zweijähriger Grundkurs und als Leistungskurs in der Kollegstufe, also mit zusätzlich drei bzw. fünf Wochenstunden Unterricht in den beiden letzten Schuljahren, etabliert wird.

Diese organisatorischen Änderungen an den Gymnasien haben Folgen für die Ausbildung der Informatiklehrer: Die Erweiterungsprüfung als einzige Möglichkeit, die Lehrbefähigung in Informatik zu erlangen, reicht nicht mehr aus; die Informatik kann ab 2003, voraussichtlich in Kombination mit Physik oder Wirtschaftswissenschaften, als grundständiges Fach für das Lehramt an höheren Schulen auch in Bayern studiert werden. Man hofft, dass damit auch mehr Studenten die Lehrbefähigung in Informatik erwerben wollen, denn trotz der Kompaktkurse in München und Erlangen für bereits in Beruf stehende Lehrer gibt es für die etwa 400 Gymnasien kaum mehr als 100 Lehrer mit dieser eingeschränkten Lehrbefähigung in Informatik, siehe z. B. [1].

Für das grundständige Studium müssen nicht nur die bewährten Vorlesungen für den Diplomstudiengang an diese neue Ausbildungsrichtung angepasst werden. Die Didaktik der Informatik erhält auch einen neuen Stellenwert. Bisher wurde sie in Ergänzung zu den Fachdidaktiken der anderen Lehramtsfächer der Staatsexamenskandidaten abgehalten; ihre Belegung war ebensowenig wie die der Fachvorlesungen zur Erweiterungsprüfung nachzuweisen. Mit dem grundständigen Studium müssen Pflicht-

vorlesungen, Übungen und Seminare in Didaktik eingeführt werden, so z. B. auch für das studienbegleitende Praktikum.

Parallel zu den organisatorischen Änderungen fand auch eine grundlegende Neuorientierung der Inhalte der Schulinformatik statt, siehe z. B. [2] und [4]. Während früher unter einem Informatikunterricht in erster Linie ein Programmierkurs verstanden worden ist, sind jetzt Modellierungen und Prinzipien bei der Informationsverarbeitung zentrale Themen. Lehrpläne dazu sind für Versuchsschulen bereits aufgestellt [3].

Aus der veränderten Situation ergeben sich Forderungen für die Stellen, die für die Didaktik der Informatik an den Universitäten verantwortlich sind. Zum einen müssen sie die Grundlagen lehren, die bei der Umsetzung der Inhalte in einen Unterrichtsablauf wichtig sind. Dabei ist in einem viel höherem Maß Entwicklungsarbeit zu leisten, als dies bei anderen Fachdidaktiken der Fall ist; bei der Ausbildung der Studenten wird hier in weiten Bereichen Neuland betreten.

Zum anderen hat ein enger Kontakt zu den Schulen einen höheren Stellenwert als bei den Fachdidaktiken in den anderen Lehramtsfächern. Allgemein kann man keine gültigen Aussagen über das Unterrichtsgeschehen treffen, ohne die aktuellen Erziehungsprobleme genau zu kennen; aus diesem Grund sollten die Vertreter der Didaktik auch regelmäßig in Schulen unterrichten. In der Informatik tritt bei dem Schulkontakt eine weitere Aufgabe in den Vordergrund: Er ist der notwendige Rahmen, das Erstellen und Ausprobieren von neuen Unterrichtsmodellen beratend zu begleiten, um den beteiligten Lehrkräften zu helfen, denn diese beschreiten ja dabei auch zum großen Teil Neuland. Ebenso sind die Didaktiker bei der Koordination und Mithilfe von Lehrerfortbildungen und bei der Beratung der Lehrkräfte, die Informatik unterrichten müssen, aber keine Ausbildung dazu haben, besonders gefordert.

Die Didaktik der Informatik kann nicht wie die Fachdidaktiken der anderen Lehramtsfächer auf eine langjährige Grundlagenforschung zurückgreifen. Auch hier besteht ein Nachholbedarf, der nur dann geleistet werden kann, wenn sie genügend fest an der Universität verankert ist.

[1] http://www2.informatik.uni-erlangen.de/IMMD-II/Persons/Guests/kreisel/vortrag300499.html).

[2] Hubwieser P., Broy M.: „Der informationszentrierte Ansatz. Ein Vorschlag für eine zeitgemäße Form des Informatikunterrichts am Gymnasium." Technischer Bericht der TU München, TUM-I9624, Mai 1996.

[3] http://www.isb.bayern.de/bf/isbl/lps/gym/informatik.htm (für das Europäische Gymnasium, Typ III)

[4] Zentralstelle für Computer im Unterricht (Hg.): "Informatik in der Schule", Augusburg 2000

(Kreisel)

6 DiaGen—Spezifikation und Generierung graphischer Diagrammeditoren

Diagramme werden überall verwendet, wo komplizierte Sachverhalte verständlich dargestellt werden sollen. Beispiele sind Notenblätter in der Musik, elektronische Schaltpläne in der Elektrotechnik oder UML-Diagramme in der Informatik. Diagrammeditoren sind graphische Editoren, die auf eine bestimmte Diagrammsprache zugeschnitten sind. Als Bestandteil größerer Software-Systeme können sie die Kommunikation mit dem Benutzer an Hand von Diagrammen ermöglichen. Sie können aber auch eigenständige Programme sein, mit denen Diagramme gezeichnet und manipuliert werden. Von Zeichenprogrammen, die beispielsweise Bestandteil der heute gebräuchlichen Office-Pakete sind, unterscheiden sich Diagrammeditoren dadurch, dass sie einerseits die damit erstellten Diagramme und ihre Bedeutung zu einem gewissen Grad "verstehen". Andererseits kann man mit Diagrammeditoren nicht beliebige Zeichnungen oder Diagramme erzeugen; vielmehr wird der Anwender auf die Verwendung solcher Diagrammkomponenten beschränkt, die in der Diagrammsprache vorkommen, auf die der Editor zugeschnitten ist. So wird man in einem UML-Klassendiagrammeditor im allgemeinen kein Transistorsymbol verwenden können, das ein Editor für elektronische Schaltpläne sicher in seinem Repertoire hat.

Prinzipiell lassen sich zwei Editiermodi unterscheiden, die die Interaktion des Benutzers mit dem Diagrammeditor bestimmen: Beim *strukturierten* oder *syntaxgesteuerten* Editieren stehen Operationen zur Verfügung, mit denen der Benutzer komplexe Diagrammänderungen vornehmen kann. Diese Operationen ändern in erster Linie die Bedeutung eines Diagramms, dessen graphische Darstellung dann vom Editor entsprechend angepasst wird. Wie mit einem Zeichenprogramm kann der Benutzer dagegen beim *freien* Editieren das Diagramm und seine graphische Darstellung unmittelbar verändern. Da auf diese Weise auch Diagramme entstehen können, die nicht den Regeln der vorgegebenen Diagrammsprache entsprechen, muss der Editor die Korrektheit des Diagramms prüfen und den Benutzer gegebenenfalls auf Unstimmigkeiten hinweisen. Darüber hinaus muss er ausgehend von der graphischen Darstellung des Diagramms dessen Bedeutung ermitteln. Voraussetzung hierfür ist eine geeignete Formalisierung (eine sogenannte *Grammatik*) der Diagrammsprache und deren Regeln.

Beim Vergleich der beiden Editiermodi spricht zu Gunsten des strukturierten Editierens, dass dem Benutzer Editieroperationen zum komplexen Ändern von Diagrammen an die Hand gegeben werden, während sich der Editornutzer beim freien Editieren selbst um die korrekte graphische Darstellung des Diagramms gemäß den Regeln der Diagrammsprache kümmern muss. Für den freien Editiermodus spricht dagegen, dass dem Benutzer alle Freiheiten gelassen werden und er nicht auf die Verwendung einer vorgegebenen Menge von Editieroperationen beschränkt ist, die ihm einen bestimmten Editierstil aufzwingen.

Diagrammeditoren haben bislang entweder den einen oder den anderen Editiermodus unterstützt, nicht aber gleichzeitig beide Modi, obwohl solche Editoren die Vorteile beider Modi kombinieren und deren

jeweiligen Nachteile ausgleichen würden. Ferner war die Programmierung eines Diagrammeditors bislang im allgemeinen sehr aufwendig, da es nur wenige Ansätze zur Automatisierung gab. Im letzten Jahr wurde die Forschung in diesem Themengebiet vorangetrieben, um die angesprochenen Defizite zu beheben. Es wurde ein Verfahren erarbeitet, mit dem beliebige Diagrammsprachen formal spezifiziert und Diagrammeditoren aus solchen Spezifikationen weitgehend automatisch generiert werden können. Der für die Realisierung eines Diagrammeditors nötige Aufwand wird dadurch drastisch reduziert. Gegenüber bereits existierenden Ansätzen hat die beschriebene Methode außerdem den Vorteil, dass Diagramme mit den so erzeugten Editoren sowohl strukturiert als auch frei editiert werden können.

Das entwickelte Verfahren basiert auf der Modellierung von Diagrammen mit Hypergraphen und der Spezifikation der Diagrammsprache an Hand einer Hypergraphgrammatik. Freies Editieren wird dadurch unterstützt, dass ein mit dem Zeicheneditor frei erstelltes Diagramm durch ein äquivalentes Hypergraphmodell repräsentiert und dieses — analog zu klassischen Compilertechnik für textuelle Programmiersprachen — lexikalisch, syntaktisch und semantisch analysiert wird. Der Editor überprüft dabei die Korrektheit des Diagramms, zeigt dem Benutzer Unstimmigkeiten an und ermittelt die syntaktische Struktur des Diagramms. Um die im Diagramm ausgedrückte Information nutzen zu können, übersetzt der Editor das Diagramm in Datenstrukturen, die die Bedeutung des Diagramms wiedergeben. Im Mittelpunkt dieser Analyse steht ein Hypergraphparser, der die syntaktische Struktur des Hypergraphmodells in Bezug auf die spezifizierte Hypergraphgrammatik analysiert, sich dabei robust gegenüber syntaktischen Fehlern im Diagramm bzw. seinem Hypergraphmodell zeigt und solche Fehler identifizieren kann. Strukturiertes Editieren wird durch Editieroperationen ermöglicht, die das Hypergraphmodell eines Diagramms unmittelbar verändern. Solche Operationen werden mit programmierten Hypergraphtransformationen spezifiziert. Ein automatischer Layoutmechanismus, der die vom Hypergraphparser ermittelte syntaktische Struktur des Diagramms berücksichtigt und mit graphischen Constraints deklarativ spezifiziert werden kann, sorgt für die korrekte graphische Darstellung so editierter Diagramme.

Dieses Verfahren bildet die Grundlage für ein *DiaGen* in Java realisiertes *Rapid Prototyping*-Werkzeug zur Generierung von Diagrammeditoren, die als Software-Komponenten unmittelbar in andere Software-Systeme integriert werden können. Mit *DiaGen* wurde bereits eine Vielzahl von Diagrammeditoren generiert, darunter Editoren für UML-Klassendiagramme, Flussdiagramme und Struktogramme im Bereich der Informatik sowie Petrinetze und Kontaktpläne im Bereich der Steuerungstechnik. Damit konnte gezeigt werden, daß das in dieser Arbeit vorgestellte Verfahren praktisch einsetzbar ist. Insbesondere haben sich Hypergraphparser als effektives und vor allem effizientes Hilfsmittel zur Diagrammanalyse erwiesen.

Nähere Informationen sind unter: http://www2.informatik.uni-erlangen.de/DiaGen zu finden.

(Minas)

7 Netzgestützter Lehrverbund zur Lehrerausbildung in Informatik (NELLI)

Im Rahmen des Projektes NELLI entsteht ein speziell zugeschnittenes und modular aufgebautes Angebot an internetbasiertem Studienmaterial für die Aus- und Weiterbildung von Lehrkräften in Informatik. An dem Projekt sind die FAU Erlangen, die LMU München, TU München und die Univ. Passau beteiligt. In Erlangen wird ein Lernmodul zur Algorithmik entwickelt.

Die Grundlage des Algorithmik-Moduls bildet die Vorlesung Algorithmik I, die seit 1992 an der Friedrich-Alexander-Universität Erlangen-Nürnberg für die Studierenden des Faches Informatik im 1. Semester angeboten wird. Diese Vorlesung wird auch von Wirtschaftsinformatikern, Computerlinguisten und Mathematikern mit Nebenfach Informatik sowie von Studierenden einiger anderer Fächer gehört. Die Vorlesung hat einen Umfang von 4 Semesterwochenstunden, dazu werden Tafelübungen und betreute Rechnerübungen im Umfang von jeweils 2 SWS angeboten. Der Inhalt der Vorlesung basiert weitgehend auf dem Lehrbuch „Structure and Interpretation of Computer Programs“ von Abelson und Sussman, das seit Jahren am MIT in der Ausbildung eingesetzt wird. Darüber hinaus werden theoretische Konzepte eingeführt, welche die Grundlage der systematischen Entwicklung von Software bilden. Diese Ergänzungen zur theoretischen Informatik sind in dem Buch „Informatik - Eine grundlegende Einführung“ von Broy zu finden. Mit der Lehrveranstaltung soll vermittelt werden, dass bei der Entwicklung von komplexer und verlässlicher Software eine ingenieurmäßige, theoretisch fundierte Vorgehensweise unerlässlich ist. Dabei zeigt sich, dass die Beschäftigung mit den theoretischen Grundlagen von Algorithmen auf natürliche Art die Denkweise schult, die für das systematische, praktische Arbeiten am Computer notwendig ist. Als Programmiersprache wird der LISP-Dialekt Scheme eingesetzt.

Den Ausgangspunkt für die Erstellung des Lehrmaterials bilden die Folien zu der Vorlesung, die im LaTeX-Format vorliegen. Diese Folien enthalten den Vorlesungsstoff jedoch naturgemäß nur stichpunktartig. Die für das Verständnis unerlässlichen, wesentlichen Erläuterungen, die in der Vorlesung vermittelt werden, müssen ergänzt werden, damit die Inhalte auch ohne Teilnahme an der Vorlesung nachvollziehbar werden. Darüber hinaus enthält der Stoff viele Beispiele, bei denen komplexe Strukturen schrittweise entwickelt werden. Diese Entwicklung soll zusammen mit den Erläuterungen ebenso in einer vom Benutzer des multimedialen Lehrmaterials interaktiv zu bestimmenden Geschwindigkeit dargestellt werden können. Das Lehrmaterial wird nun in die XML-basierte Sprache LMML umgeschrieben. Diese Sprache wird von den Passauer Projektpartnern entwickelt und auch von den Münchner Projektpartnern verwendet, so dass nun alle Nelli-Module in einem einheitlichen Format erstellt werden. Da die zur Zeit verbreiteten Internet-Browser, die in XML erstellten Dokumente noch nicht (Netscape Communicator) oder nur nach Modifikationen (MS Internet-Explorer) darstellen können, müssen die XML-Dokumente ins HTML-Format übersetzt werden. Ein entsprechend konfigurierter

Webserver wurde auch in Erlangen installiert.

Für die praktischen Programmierübungen kommt das System „DrScheme“ zum Einsatz, das an der Rice-University in Texas entwickelt wurde. Diese Programmierumgebung ist für alle gängigen Plattformen kostenlos erhältlich und zeichnet sich durch eine intuitive Bedienung auf Basis einer grafischen Oberfläche aus. Das System wurde im Sommersemester 2000 im Rahmen eines Tutoriums zur Algorithmik-Vorlesung den Studierenden vorgestellt, die „DrScheme“ als sehr gut beurteilten. Ein weiterer Vorzug dieses Systems stellt die Integration von grafischen Elementen bei der Programmierung dar. Konzepte wie z. B. die Rekursion lassen sich so ansprechend visualisieren, was sich sehr motivationsfördernd auswirkt.

Seit September 2000 arbeitet Herr Franz Forman am Algorithmik-Modul mit. Er hat als Lehrer für Informatik am Gymnasium bereits Kurse auf Basis des LISP-Dialekts LOGO konzipiert und gehalten und überträgt nun auch entsprechende Programme von LOGO nach Scheme, so dass im Algorithmik-Modul auch Programmierbeispiele vorgestellt werden, die im Unterricht am Gymnasium weiterverwendet werden können.

Es ist geplant, im Frühjahr das bis dahin in LMML vorliegende Material in Erlangen bei einer Lehrerfortbildung interessierten Lehrkräften vorzustellen und dann im Rahmen eines Probebetriebes zu testen.

(Schmidt)

8 Schichtplanung mit HybrideN Genetischen Algorithmen

Ein wichtiger Faktor beim Management von Ressourcen eines Unternehmens ist die Verwendung der menschlichen Arbeit. Insbesondere in Situationen mit variierenden Besetzungsstärkenanfoderungen, unterschiedlichen Qualifikationen der Mitarbeiter und unterschiedlichen Arbeitsverträgen können starre Einteilungsschemata nicht verwendet werden. Daher sind flexible automatische Planungsmethoden notwendig, um den menschlichen Planer zu unterstützen. Wir haben gezeigt, wie Genetische Algorithmen, verbessert durch die Anwendung problemspezifischen Wissens, angewendet werden können, um derartige Planungsprobleme zu lösen.

(Gröbner/Wilke)

9 Risikomanagement

Aufgrund des im März 1998 verabschiedeten Gesetzes zur Kontrolle und Transparenz im Unternehmensbereich (KonTraG) sind Aktiengesellschaften verpflichtet, ein Risikomanagementsystem und ein Frühwarnsystem zu installieren. Die Verantwortung liegt direkt beim Vorstand. Ein vom Land

Bayern gefördertes Projekt hat die Entwicklung eines software-unterstützten Risikomanagementsystems mit integriertem Frühwarnsystem zum Ziel. Das zu erstellende System soll den Geschäftsprozess Risikomanagement modellieren, der interne Geschäftsablauf für diese Aufgabe optimiert und seine Durchführung vereinfacht werden.

Zentrales Element ist die Entscheidungsunterstützungskomponente, deren Aufgaben ist:

- Identifikation von Risiken

- Risikobewertung

- Handhabung der risikomindernden und risikovermeidenden Maßnahmen

- Controlling für das Risikomanagement

Das Projekt wurde Ende des Jahres von der Bayerischen Staatskanzlei genehmigt und hat eine Laufzeit von 2 Jahren.

(Wilke/Gröbner)

10 Flexible drahtlose Kommunikation für mobile Roboter

Kommunikation ist eine wichtige Voraussetzung für kooperatives Verhalten von intelligenten Robotern in einem Multi-Agenten-System. Wir haben dazu die Kommunikation zwischen individuelle Roboter, die RoboCup spielen, beschrieben. Diese Roboter basieren auf der EyeBot-Plattform, siehe: http://www.ee.uwa.edu.au/~braunl/eyebot.

Die aus der Literatur bekannten Ansätze wurden auf ihre Verwendbarkeit für diese Aufgabe untersucht und eine Konzeption erarbeitet, die das Nachrichten-Format, die selbständige Konfiguration und den Wiederanlauf nach einem Fehler einschliesst. Die Kommunikation erlaubt die Übertragung von Nachrichten zwischen individuellen Robotern, Rechnern und die Sendung an alle Kommunikations-partner (broadcast).

(Wilke)

Habilitationen:

- Minas, Mark, Dr.-Ing.: *"Spezifikation und Generierung graphischer Diagrammeditoren"*

Studienarbeiten:

- Andres, Anna-Maria: *Rechnen mit DNA Studienarbeit zum Themabereich DNA Computing"*

- Fühner, Tim: *"Visualisierung evolutionärer Algorithmen im Rahmen des Evolutionssystems EVOLVICA"*
- Köth, Oliver: *"Entwurf und Implementierung eines generisches graphischen Editors"*
- Lörcher, Günther: *"Implementierung einer Benutzeroberfläche zur Parametrierung einer Netzwerkkomponente"*
- Milovanovic, Igor: *"Entwurf und Implementierung einer Knowlegde-Base im Rahmen des Qualitätsmanagementes"*
- Weigand, Christian: *"Entwurf und Entwicklung von Active-X-Komponenten, lauffähig unter Windows CE R und Windows 95/98 sowie NT 4.0"*

Diplomarbeiten:

- Bloß, Jürgen: *"Entwurf und Implementierung eines Client/Server-Systems für ein Versionsarchiv"*
- Fuchs, Matthias: *"Entwurf und Implementierung eines Formeleditors zum Einsatz im Schulbereich"*
- Gast, Daniel: *"XML-basierte Definition und Verwaltung von Software-Entwicklungsprozessen"*
- Gericke, Jörg: *"Implementation eines verteilten Versionsarchivs"*
- Haselmann, Ralf: *"Designing and Implementing the basic structure, genetic operators and generation of the initial populations in a learning system combining ILG and GA"*
- Sander, Knut: *"Designing and Implementing the transformation, fitness and selection mechanism in a learning system combing ILP and GA"*

Veröffentlichungen:

Chang, N., Fischer, I.: *"Modeling Idioms with Constructions, Metaphors and Simulation"*. 22. Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft, Philipps-Universität Marburg, AG 7: Kollokationen, linguistische Beschreibung und Akquisition aus Texten, 1.-3. März 2000

Chang, N., Fischer I.: *"Understanding Idioms"*.
Konvens 2000/Sprachkommunikation, 5. Konferenz zur Verarbeitung natürlicher Sprache - ITG-Fachtagung "Sprachkommunikation", Technische Universität Ilmenau, 9.- 12. Oktober 2000

Eitschberger, M., Enzelberger, S., Steuss, K., Roth, M., Frank, C., Fischer, I.: Mädchen + Technik Praktikum 2000, Technisch-Naturwissenschaftliches Schnupperpraktikum für Schülerinnen Fraunhofer IRB Verlag, Stuttgart, ISBN 3-8167-5611-5, 2000

Fischer, I.: Koch, M., Berthold, M. R.: *"Attributed Graph Transformation with Partial Attribution*". In H. Ehrig, G. Taentzer (eds.) Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems, Technischer Bericht der Technischen Universität Berlin, Nummer 2000-2, ISSN 1436-9915, 25 - 27 März 2000, S. 171 - 178

Fischer, I.: Koch, M., Berthold, M. R.: *"Learning and Rewriting in Fuzzy Rule Nets in Proceedings Applications of Graph Transformation With Industrial Relevance"*. (AGTIVE '99), Kerkrade, Niederlande, September 1999, 2000, In LNCS 1779, S. 263 - 270,

Fischer, I., Koch, M., Taenzer, G.: *"Local Views on Distributed Systems and their Communications"* In H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg: Theory and Application of Graph Transformation, LNCS 1764, 2000, S. 162-176

Frey, G. and Minas, M.: *"Editing, Visualizing, and Implementing Signal Interpreted Petri Nets"*. In Proc. 7. Workshop Algorithmen und Werkzeuge für Petrinetze (AWPN'2000), S. 57-62,

TR 7/2000 Fachberichte INFORMATIK Universität Koblenz-Landau; Institut für Informatik Koblenz, 2. - 3. Oktober 2000

Haselmann, R., Kókai, G., Sander, K.: *"Ein genetisch logisches Programmiersystem"*. In Proc FGML-2000 Treffen der GI-Fachgruppe 1.1.3 Maschinelles Lernen, GMD Report 114, S. 7-12 Bonn, Germany, 18. -20. September 2000

Hoffmann, B., Minas, M.: *"Über generisches visuelles Programmieren"*. In S. Diehl und A. Kerren (eds.), Tagungsband des GI-Workshops *"Softwarevisualisierung 2000"*, Schloß Dagstuhl, 11.-12. Mai 2000, TR A/01/2000, Universität des Saarlandes, Saarbrücken, Fachbereich Informatik, S. 41-51

Hoffmann, B., Minas, M.: *"A Generic Model for Diagram Syntax and Semantics"*. Workshop on Graph Transformation and Visual Modelling Techniques (GT-VMT 2000). 15. - 16. Juli 2000, Genova, Switzerland. In: J.D.P. Rolim et al (eds.): ICALP Workshops 2000, Proceedings in Informatics 8, Carleton Scientific, Waterloo, Ontario, Canada, S. 443-450, 2000

Hoffmann, B., Minas, M.*: "Towards Generic Rule-Based Visual Programming"*. In Proc. 2000 IEEE Symposium on Visual Languages (VL'2000), Seattle, Washington, S. 65f, IEEE Computer Society Press, 2000

Hoffmann, B., Minas, M.: *"Towards Rule-Based Visual Programming of Generic Visual Systems"*. In Proc. First International Workshop on Rule-Based Programming (RULE'2000) Montreal, Canada, 19. September 2000, S. 111-125

Köth, O., Minas, M.: *"Generating diagram editors providing free-hand editing as well as syntax directed editing"*.
In H. Ehrig, G. Taentzer (eds.) Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems, Technischer Bericht der Technischen Universität Berlin, 25 - 27. März 2000, Nummer 2000-2, ISSN 1436-9915, S. 32-39

Kókai, G.: *"New Missing Solution Method for Trace Tree"*.
In Proc. 4. WLP - Workshop Logische Programmierung, GMD Report 90, S. 199-209, Julius-Maximilians-Universität Würzburg, 26. - 28. Januar 2000

Kókai, G.: *"CAAP: Statische Codeanalyse von PROLOG Programmen"*.
In Proc. 4. WLP - Workshop Logische Programmierung GMD Report 90, S. 231-240, Julius-Maximilians-Universität Würzburg, 26. - 28. Januar 2000

Kókai, G., Ványi, R., Tóth Z., Petõ, T.: *"Parametric L-System Description of the Retina with Combined Evolutionary Operators*".
The 6th Symposium on Computers in Diabetes (CID2000), Herzliya, Israel, 14. - 16. September 2000. In the Journal Diabetes, Nutrition and Metabolism Clinical and Experimental Vol 13, N. 4.240

Minas, M. and Köth, O.: *"Generating Diagram Editors with DiaGen"*.
In Proc. of the International Workshop on Applications of Graph Transformation with Industrial Relevance (AGTIVE'99) at Monastery Rolduc, NL, 1. - 3. September 1999, LNCS 1779, Springer-Verlag 2000, S. 433-440

Minas, M.: *"Creating semantic representations of diagrams"*.
In Proc. of the International Workshop on Applications of Graph Transformation with Industrial Relevance (AGTIVE'99) at Monastery Rolduc, NL, 1. - 3. September 1999, LNCS 1779, Springer-Verlag 2000, S. 209-224

Minas, M: *"Hypergraphs as a Uniform Diagram Representation Model"*.
In Proc. 6th International Workshop on Theory and Application of Graph Transformations (TAGT'98), Paderborn, Germany, November 1998. LNCS 1764, Springer-Verlag 2000, S. 281-295

Schneider, H.J.: *"The joy of teaching theoretical computer science"*.
Bull. Europ. Assoc. Theoretical Computer Science 70 (2000), pp. 165-166

Schneider, H.J.: "*Graph transformations - Where they come from and where they should go".*
In H. Ehrig, G. Taentzer (eds.) Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems, Technischer Bericht der Technischen Universität Berlin, Nummer 2000-2, ISSN 1436-9915, pp. 270-271, 25 - 27 März 2000

Schneider, H.J.: *"Computability in an introductory course on programming".*
Bull. Europ. Assoc. Theoretical Computer Science (To appear)

Tóth, Z., Kókai, G., Ványi, R.: *"Interactive Visual Tree Evolution".*
In Proc. EIS2000 Second International, ICSC Symposium on engineering of intelligent systems, University of Paisley, Scotland, U.K., 27. - 30. Juni 2000

Ványi, R., Kókai, G., Tóth Z., Petõ, T.: *"Grammatical Retina Description with Enhanced Methods".*
In Proc. EuroGP2000 third european conference on genetic programming, Edinburgh, 15.- 16. April 2000, LNCS 1802, S. 193-209

Wilke, P.: *"Time Series Prediction for Workforce Management Systems for Call Centres".*
Second International ICSC Symposium on Neural Computation NC 2000 Program, 23. -26. Mai 2000, Berlin, Deutschland, Buch und CD Rom, S. 45ff

Gastvorträge:

Csendes, Tibor: *"Circle packing in the unit square"*

Grabska, Ewa: *"From Graph operations to graph rules in CAD"*

Horváth, Tamás: *"Learning Logic Programs with Structured Background Knowledge"*

Slusarek Maciej: *"Reduction of chain complexes: a new approach to computational problems in algebraic topology"*

Vorträge:

Fischer, I.: 22. *"Modeling Idioms with Constructions, Metaphors and Simulation".*
Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft, Kollokationen, linguisti sche Beschreibung und Akquisition aus Texten, Philipps-Universität Marburg, AG 7, 1. März 2000

Fischer, I.: *"Understanding Idioms".*
KONVENS 2000/Sprachkommunikation, 5. Konferenz zur Verarbeitung natürlicher Spra che - ITG-Fachtagung "Sprachkommunikation", Technische Universität Ilmenau, 9. Okto ber 2000

Fischer, I.: "*Attributed Graph Transformation with Partial Attribution"*.
Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems, GraTra'2000", Technische Universität Berlin, 26. März 2000

König R.: *"Teoria dei Codici"*.
Lehraufenthalt im Rahmen des Sokrates-Erasmus Programms für Studenten der Mathematik und der Informatik, Università degli studi di Cagliari, Italien, 2. - 30. April 2000

Kókai, G.: *"New Missing Solution Method for Trace Tree"*.
4. WLP - Workshop Logische Programmierung, Würzburg, 27 Januar 2000

Kókai, G.: *"CAAP: Statische Codeanalyse von PROLOG Programmen"*.
4. WLP - Workshop Logische Programmierung, Würzburg, 28. Januar 2000

Kókai, G.: *"Grammatical Retina Description with Enhanced Methods"*.
EuroGP2000, third european conference on genetic programming, Edinburgh, 15. April 2000

Kókai, G.: *"Parametric L-System Description of the Retina with Combined Evolutionary Operators*".
The 6th Symposium on Computers in Diabetes (CID'2000), Herzliya, Israel, 15. September 2000

Kókai, G.: *"Ein genetisch logi sches Programmiersystem"*.
FGML-2000 Treffen der GI-Fachgruppe 1.1.3 Maschinelles Lernen, Bonn, Germany, 18. September 2000

Minas, M.: *"Generating Diagram Editors Providing Free-Hans Editing al well al Syntax-Directed Editing"*.
Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems, GraTra'2000, Berlin, 25. März 2000

Minas, M.: "*DiaGen-Systemdemo"*.
Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems GraTra'2000, Berlin, 26. März 2000

Minas, M.: "*DiaGen-Systemdemo"*.
Workshop SV'2000, Dagstuhl, 11. Mai 2000

Minas, M.: *"Automatisierte Erstellung graphischer Editoren"*.
Firmenkolloquium der Firma infoteam, Bubenreuth, 30. Mai 2000

Minas, M.: *"Towards Generic Rule-Based Visual Programming"*.
Symposium VL'2000, Seattle, 11. September 2000

Minas, M.: *"Towards Rule-Based Visual Programming of Generic Visual Systems"*.
Worksop RULE'2000, Montreal, 19. September 2000

Minas, M.: *"Automatisierte Erstellung graphischer Editoren: Ein Editor für UML-Klassendiagramme in einem Tag"*.
Betriebsseminar der Firma Basys GmbH, Langlau, 17. November 2000

Minas, M.: *"Objektorientierte Umsetzung kategorientheoretischer Konzepte"*.
Habilitationsvortrag, Technische Fakultät der FAU Erlangen, 29. November 2000

Wilke, P.: *"Time Series Prediction for Workforce Management Systems for Call Centres*".
Second International ICSC Symposium on Neural Computation" NC 2000 Program,
23. - 26. Mai 2000, Berlin, Buch und CD Rom, S. 45f