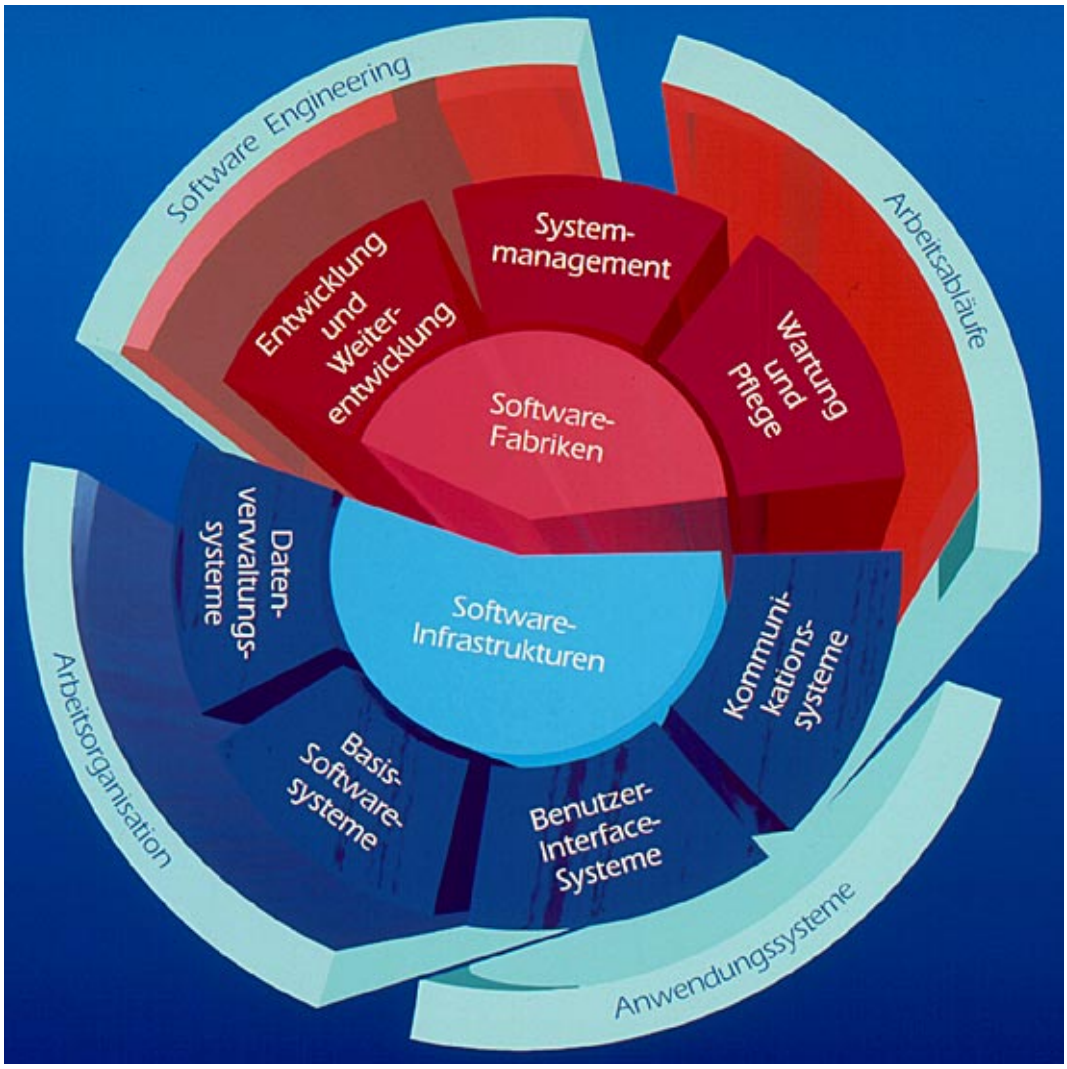
Fraunhofer-Einrichtung
für Software-
und Systemtechnik



Tätigkeitsbericht 1992

Tätigkeitsbericht 1992

Fraunhofer-Einrichtung
für Software- und Systemtechnik ISST

Vorwort

Das Institut für Software- und Systemtechnik (ISST) Berlin und Dortmund ist eine der neuen Einrichtungen der Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V. Es wurde gegründet, um die Informatikforschung innerhalb der Fraunhofer-Gesellschaft zu stärken.

In informationstechnischen Systemen gewinnt Software gegenüber der Hardware zunehmend an Bedeutung. Im Gegensatz zur modernen industriellen Fertigung ist die Softwareproduktion heute jedoch noch weitgehend durch unsystematische Einzelfertigung geprägt. Die Industrialisierung der Entwicklung von Software und deren Einbettung in *integrierte informationstechnische Infrastrukturen* sind deshalb die vorrangigen Ziele zeitgemäßer Software- und Systemtechnik.

Das ISST hat die Aufgabe, systematische, ingenieurtechnische Vorgehensweisen des Software Engineering zu entwickeln und zu fördern sowie an der breiten Einführung dieser Techniken in die industrielle Anwendung im Rahmen integrierter informationstechnischer Infrastrukturen mitzuwirken.

Dabei kann das ISST auf wesentlichen Vorarbeiten aufbauen, die am Lehrstuhl für Softwaretechnologie der Universität Dortmund unter meiner Leitung durchgeführt wurden.

In Berlin wie in Dortmund besteht eine enge wissenschaftliche und organisatorische Anbindung der Institutsteile an die dortigen Hochschulen (Technische Universität Berlin bzw. Universität Dortmund). In Berlin werden das ISST und das entsprechende Fachgebiet an der Technischen Universität nach einem in der Fraunhofer-Gesellschaft vielfach bewährten Muster als Doppelinstitut geführt, um so eine effiziente Verbindung der universitären Grundlagenforschung mit der am Fraunhofer-Institut angesiedelten angewandten Forschung und Entwicklung zu erreichen.

Der vorliegende Tätigkeitsbericht für das Jahr 1992 will das ISST in seinem wissenschaftlichen Profil präsentieren. Zu diesem Zwecke werden die bearbeiteten Forschungsthemen umfassend dargestellt und nicht nur kurz im Ergebnis referiert.

Prof. Dr. Herbert Weber

Inhalt

1	Übersicht	7
1.1	Gründung und Trägerschaft	7
1.1.1	Kuratorium	8
1.2	Aufgabenstellung und Arbeitsfelder	9
1.3	Leistungsangebote	12
1.4	Organisation und wissenschaftliche Bereiche	14
1.5	Personal und Sachausstattung	17
1.6	Haushalt und Finanzierung	18
2	Fachlicher Ergebnisbericht	20
2.1	Projektüberblick	20
2.2	Werkzeuggestützte Softwareentwicklung	23
2.2.1	Zielsetzung des Bereichs	23
2.2.2	Arbeitsgebiete	24
2.2.3	Speziationssprache und -Umgebung PISA	25
2.2.4	Entwicklung von LISP-Systemen	28
2.2.5	Re-Engineering	32
2.2.6	CASE-Einführung	35
2.2.7	CASE-Kompetenzzentrum	37
2.3	Integrierte Software-Infrastrukturen	38
2.3.1	Zielsetzung und konzeptionelle Grundlagen	38
2.3.2	Arbeitsgebiete	41
2.3.3	Software-Fabrik-Engineering	42
2.3.4	Daten- und Objektmanagement	45
2.3.5	Multimedia-/Hypermedia-Systeme und kooperatives Arbeiten	48
2.3.6	Systemmanagement großer Netze	50
2.4	Prozeß-Engineering	53
2.4.1	Systematisches Management von Vorgängen	53
2.4.2	Management kundenspezifischer Geschäftsprozesse	55
2.5	Aus- und Weiterbildung	56
3	Anhang	58
3.1	Veröffentlichungen	58
3.2	Veranstaltungen und Präsentationen	64
3.3	Dissertationen und Diplomarbeiten	65
3.4	Mitarbeit in Gremien, Konferenz- und Programmkomitees	66
3.5	Vorlesungstätigkeit an Hochschulen	66

1 Übersicht

1.1 Gründung und Trägerschaft

Die Fraunhofer-Einrichtung für Software- und Systemtechnik (ISST) Berlin/Dortmund ist eine Forschungseinrichtung der Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V. (FhG).

Die Fraunhofer-Gesellschaft wurde 1949 als gemeinnützige Institution gegründet; sie betreibt angewandte und anwendungsorientierte Forschung vorwiegend in den Natur- und Ingenieurwissenschaften. Der Name soll an den Physiker, Astronomen und Unternehmer Joseph von Fraunhofer (1787-1826) erinnern, der mit Erfolg versuchte, die Entwicklung neuer technischer Produkte und Produktionsverfahren auf wissenschaftlichen Theorien aufzubauen. Gegenwärtig unterhält die FhG 47 Forschungseinrichtungen mit mehr als 7500 Mitarbeitern. Das Forschungsvolumen beträgt fast eine Milliarde DM und wird zu etwa drei Vierteln aus Aufträgen der Industrie und aus öffentlichen Forschungsaufträgen erwirtschaftet. Der Sitz der Zentralverwaltung ist in München.

Das ISST wurde im Rahmen der Ausdehnung der Fraunhofer-Gesellschaft auf die neuen Bundesländer zum 1. Januar 1992 als selbständige Einrichtung in Berlin gegründet und verstärkt die Informatikforschung in der FhG. Seine Hauptaufgabe sieht es in der Entwicklung und Förderung systematischer, ingenieurtechnischer Vorgehensweisen des Software Engineering sowie in Aktivitäten zur breiten Einführung dieser Vorgehensweisen in die industrielle Anwendung im Rahmen *integrierter informationstechnischer Infrastrukturen.*

Ein Teil der Mitarbeiter des ISST Berlin kam aus den Informatik-Instituten der ehemaligen Akademie der Wissenschaften Berlin; mit Ausnahme einer Arbeitsgruppe konnte aber keine der existierenden wissenschaftlichen Arbeitsstrukturen übernommen werden.

Die wissenschaftliche Gesamtkonzeption des ISST beruht im wesentlichen auf Erfahrungen aus Forschungsarbeiten, die in den Jahren 1984 bis 1992 am Lehrstuhl für Softwaretechnologie der Universität Dortmund unter der Leitung des Institutsleiters des ISST, Prof. Dr. Herbert Weber, durchgeführt wurden.

Im Interesse einer schnellen Funktionsfähigkeit des Instituts war es daher erforderlich, Know-how nach Berlin zu transferieren. Ein umfassender Transfer von Wissen und Erfahrungen von der Universität Dortmund in das ISST Berlin war aber nicht so einfach möglich, und aufgrund der Berufung von Prof. Weber von der Universität Dortmund an die Technische Universität Berlin war der Bestand der Dortmunder Forschungsgruppe Softwaretechnologie gefährdet.

Diese Probleme konnten gelöst werden, indem eine Außenstelle in Dortmund gegründet wurde, an der heute viele der ehemaligen Mitarbeiter des Lehrstuhls für Softwaretechnologie beschäftigt sind. Für das ISST Berlin ist diese Außenstelle von großer Bedeutung, da über sie die in Nordrhein-Westfalen bestehenden industriellen Kontakte erhalten bleiben.

1.1.1 Kuratorium

Das Institut wird von einem Kuratorium beraten, dessen Mitglieder aus Wirtschaft, Wissenschaft, Politik und Verwaltung kommen und dem zur Zeit die folgenden Persönlichkeiten angehören:

Wirtschaft

- Herr Dipl. -Volkswirt H. P. Bonn
  Geschäftsführender Gesellschafter der Firma GUS mbH, Köln
  Vorsitzender des Kuratoriums
- Herr Dipl.-Ing. A. Ganser
  Direktor in der Generaldirektion Telekom, Bonn
- Herr Dipl. -Volkswirt H. Rose
  Generalbevollmächtigter IBM Deutschland GmbH

Wissenschaft

- Herr Prof. Dr. K. Kutzler
  Vizepräsident der Technischen Universität Berlin

Politik und Verwaltung

- Herr J. Stoehr
  Abteilungsleiter in der Senatsverwaltung für Wissenschaft
  und Forschung des Landes Berlin
- Herr Dr. K. Warnke-Gronau
  Ltd. MR im Ministerium für Wirtschaft, Mittelstand und Technologie des Landes NRW, Düsseldorf

Eine Erweiterung des Kuratoriums ist für die nächste Zeit angestrebt.

1.2 Aufgabenstellung und Arbeitsfelder

Die Aufgaben des ISST liegen in der angewandten Forschung und Entwicklung in den Bereichen Software Engineering und Systems Engineering für softwareintensive Systeme sowie im Technologietransfer.

Software nimmt heute in weiten Bereichen eine Schlüsselstellung ein. Sie bildet in Informations- und Kommunikationssystemen den Hauptbestandteil und ist das entscheidende Bindeglied zwischen den Komponenten in komplexen technischen Systemen. Softwaresysteme sind langlebige Wirtschaftsgüter; wegen ihres immateriellen Charakters wird aber die Bedeutung von Software häufig unterschätzt. In letzter Zeit wird jedoch immer offensichtlicher, daß einschneidende ökonomische Nachteile und Risiken dadurch entstehen, daß benötigte Software nicht verfügbar oder die verwendete Software von unzureichender Qualität ist.

In eklatantem Widerspruch zur Bedeutung von Software stehen die gemeinhin beobachtete Qualität der Software und die Art, wie Software heute noch produziert wird. Heutige Software ist häufig fehlerhaft, unzuverlässig, unsicher und schwer verständlich. In noch stärkerem Maße trifft dies auf ältere Softwaresysteme zu, die ungeachtet dessen in der Praxis unverzichtbar sind und einen bedeutenden Investitionswert (weltweit mehrere Hundert Milliarden Dollar) darstellen. Ihre Wartung wird immer aufwendiger und bindet gegenwärtig 60 bis 80% der gesamten für Softwareentwicklung und -wartung verfügbaren Kapazität. Im Gegensatz zur Fertigung von Rechnerhardware und anderen technischen Produkten ist die Produktivität bei der Herstellung von Software in den vergangenen 30 Jahren kaum gestiegen. Software wird heute noch - wie vor 30 Jahren - zu großen Teilen unsystematisch hergestellt.

Diese Erscheinungen machen die sogenannte Softwarekrise aus, die sich als Qualitäts-, Wartungs- und Produktivitätskrise manifestiert.

Die Ursachen dieser Krise liegen im wesentlichen darin begründet, daß die Herstellung von Software nicht in dem notwendigen Maße nach ingenieurwissenschaftlichen Prinzipien betrieben, sondern noch weitgehend als „Kunst“ betrachtet wird. Das zeigt sich in der ungenügenden Verwendung formaler Beschreibungen und Methoden, in der zu schwachen Gewichtung der Spezifikations- und Entwurfsphase, in der fehlenden Systematisierung, Modellierung und Einhaltung des sinnvollen Vorgehens, in dem großen Anteil von „Handarbeit“ und in der ungenügenden Verwendung von Software-Entwicklungswerkzeugen. Wiederverwendung vorhandener Software wird kaum praktiziert, obwohl nur ein kleiner Teil der durch neue Software repräsentierten Problemlösungen auch wirklich neu ist.

Durch die zunehmende Vernetzung von Rechnern unterschiedlicher Art wird auch die Integration der auf diesen Rechnern laufenden Anwendungen nötig, um anwendungsgerechte Gesamtlösungen sicherzustellen. So entstehen aus zunächst autonomen kommerziellen Anwendungssystemen, Dispositions- und Entscheidungsunterstützungssystemen, Bürokommunikations- und Dokumentenverarbeitungssystemen, Führungs- und Leitsystemen, Multimediasystemen, Datenhaltungs- und

Datenverwaltungssystemen sowie Kommunikationssystemen hochkomplexe integrierte Systeme, in denen vielfältige Wechselwirkungen zwischen den Einzelsystemen auftreten. Solche Systemgeflechte werden in Analogie zu komplexen technischen Geflechten anderer Art „Informationstechnische Infrastrukturen“ genannt. Diese bekommen alle Fähigkeiten zum Erfassen, Transportieren und Verarbeiten von Informationen.

Mit der Zusammenführung von Einzelsystemen zu informationstechnischen Infrastrukturen vollzieht sich ein drastischer Wandel der Natur solcher Systeme. Sie werden nicht mehr als Produkte für einen begrenzten Zeitraum, sondern für eine im Prinzip unbegrenzte Lebensdauer entwickelt. Die notwendigen Weiterentwicklungen, Anpassungen und Erweiterungen erfolgen durch partielle Erneuerung, partielle Erweiterung und durch partiellen Ersatz, ohne daß die Grundkonzeption der Infrastruktur in Frage gestellt wird. Informationstechnische Infrastrukturen sind heute in Ansätzen erkennbar und werden am Ende dieses Jahrzehnts weltweit existieren und staatliche Organisationen, Unternehmen und auch Privathaushalte umfassen.

Informationstechnische Infrastrukturen erfordern eine Integration sowohl von Hardware- als auch von Softwaresystemen. Während die Integration verschiedenster Rechnersysteme schon weitgehend beherrscht wird, stellt die vollständige Systemintegration - die sich in erster Linie auf der Ebene der Software abspielt - noch immer ein massives praktisches Problem dar, das auch wissenschaftlich kaum erforscht ist. Erforderlich sind hierfür Integrationsrahmen und -mechanismen für informationstechnische Infrastrukturen, die im wesentlichen durch *Software-Infrastrukturen* gebildet werden.

Diese „Kernsysteme“ informationstechnischer Infrastrukturen müssen z. B. auch die technologische Basis für die Realisierung der Kommunikationsbeziehungen zwischen Menschen, Werkzeugen und Plattformen oder Rechnern liefern.

Die kosteneffiziente Entwicklung, Weiterentwicklung, Pflege und Wartung von Software für informationstechnische Infrastrukturen kann nur sichergestellt werden, wenn in der Softwareentwicklung bislang unübliche Ingenieurtechniken zum Einsatz kommen, die die Verwendung standardisierter Komponenten, Plattformen und Halbfertigfabrikate vorsehen. Dies führt letztlich zu Konzepten für speziell auf die Softwareentwicklung ausgerichtete komplexe integrierte Software-Infrastrukturen, zu „Software-Fabriken“.

Leistungsfähige Software-Infrastrukturen, Software-Fabrik-Engineering und die modulare werkzeuggestützte Softwareentwicklung stellen wesentliche Ansätze für die Lösung der mit der Integration verbundenen Probleme dar und damit für die Überwindung der Softwarekrise allgemein.

Das ISST engagiert sich in den folgenden Aufgabenfeldern, wobei nahezu überall auf Ergebnisse aus dem universitären Bereich zurückgegriffen wird, die übernommen, weiterentwickelt und für den industriellen Einsatz aufbereitet werden.

- **Software Engineering**
  Die Aufgaben des Fraunhofer-Instituts liegen hier in der Aufbereitung formaler Konzepte und theoretischer Grundlagen zur Softwareentwicklung und Softwarestrukturierung. Dies betrifft z.B. formale Spezifikations- und Nachspezifikationsmethoden für zentrale, verteilte und massiv parallele Softwaresysteme, die Überführung von Spezifikationen in ausführbare Programme, Methoden der Wiederverwendung und die Leistungsanalyse und -prognose aus Spezifikationen.

- **Systems Engineering für softwareintensive Systeme**
  Die Aufgaben entsprechen denen beim Software Engineering; die betrachteten Systeme können aber neben Software- auch Hardware-, Elektronik- und mechanische Komponenten enthalten. Dies führt zu zusätzlichen komplexen Fragestellungen. Die Handhabung derartiger integrierter Systeme mit Methoden, wie sie im Software Engineering angewandt werden, ist noch weitgehend unerforscht.

- **Integrierte Software-Infrastrukturen**
  Arbeiten zur Planung, Entwicklung und Einführung integrierter Software-Infrastrukturen in die industrielle Praxis bilden für das Institut einen Schwerpunkt, um den sich viele der anderen Themen ranken. Angestrebt wird die Adaption und Entwicklung geeigneter Systemintegrationsstrategien und -technologien.
  Integrierte Software-Infrastrukturen stellen den Integrationsrahmen für komplexe (Software-)Systeme dar und bieten z.B. auch den Rahmen für moderne Software-Entwicklungsumgebungen und Software-Fabriken. Sie enthalten uniforme Benutzerinteraktionsschnittstellen, ermöglichen das Zusammenwirken von Werkzeugen, die rechnerunterstützte Zusammenarbeit von Benutzern und stellen Datenverwaltung und Prozeßmanagement zur Verfügung.

- **Datenhaltung und Datenverwaltung in verteilten Infrastrukturen**
  Eine wesentliche Aufgabe ist die Konsistenzerhaltung für unterschiedliche Datenbanksysteme, Objektverwaltungssysteme, Hybrid- und Multidatenbanksysteme.

- **Werkzeuge und Umgebungen für die Softwareentwicklung**
  Die Aufgaben bestehen in der Adaption, Instantiierung, Modifikation und Konfigurierung von Software-Entwicklungswerkzeugen (auch: „customized“ tools), in der Eigenentwicklung von Werkzeugen sowie in der Integration von Werkzeugen in Software-Entwicklungsumgebungen und integrierte Software-Infrastrukturen. Dazu gehört ebenfalls die Entwicklung geeigneter Methoden der werkzeuggestützten Softwareentwicklung.

- **Einführung der werkzeuggestützten Softwareentwicklung**
  Die Einführung der rechnergestützten Softwareentwicklung erfordert u.a. die Analyse von Software-Entwicklungspraktiken, die Formulierung der mit der Einführung angestrebten Ziele, die Erfassung der Anforderungen an Werkzeuge, deren Bewertung und Auswahl sowie die unternehmensspezifische Entwicklung von Vorgehensmodellen für Einführung und Einsatzunterstützung.

- **Integration und Re-Engineering**
  Aufgabe des Re-Engineering ist die Restaurierung/Renovierung/Sanierung (z.B. durch Restrukturierung) alter Software, die sich im praktischen Einsatz befindet und dort einen hohen Nutzen bringt, aber nicht - bzw. aufgrund vielfacher Ände-

rungen nicht mehr - die gewünschte Qualität (z.B. hinsichtlich der Wartungsfreundlichkeit) aufweist. Re-Engineering wird auch notwendig, wenn Software neuen Anforderungen genügen soll (z.B. Notwendigkeit der Portierung auf andere Rechner und Betriebssysteme, Anpassung von bzw. an Schnittstellen und an Standard-Plattformen). Re-Engineering umfaßt die Analyse und Zerlegung von Softwaresystemen in Softwarekomponenten und deren nachfolgende Zusammensetzung zu einem geänderten System sowie ggf. die Integration in neue kommerziell-technische Infrastrukturen.

- **Prozeß-Engineering**
  Wesentliches Ziel ist die Analyse und Modellierung der ablaufenden Vorgänge, die formale Überprüfung und Bewertung der Modelle und die rechnergestützte Prozeßführung. Dies führt auch zu einer Qualitätssicherung der Prozesse. Die Vorgehensweisen des systematischen Prozeßmanagements sind nicht nur für Software-Entwicklungsprozesse anwendbar, sondern weitgehend auch allgemein auf Geschäftsprozesse.
- **Qualitätssicherung**
  Qualitätssicherung betrachtet sowohl das Produkt als auch dessen Entwicklungsprozeß. Sie erfordert ein Qualitätsmodell für Softwaresysteme und Software-Infrastrukturen, ein Qualitätssicherungsmodell, Werkzeuge zur Unterstützung der Qualitätssicherung und geeignete Vorgehensmodelle.

1.3 Leistungsangebote

Die meisten der genannten Arbeitsfelder standen auch im Zentrum der Forschungen der vergangenen Jahre am Lehrstuhl für Softwaretechnologie des Fachbereichs Informatik der Universität Dortmund und einiger erfolgreicher europäischer Projekte, an denen die Dortmunder Forschungsgruppe maßgeblich beteiligt war. Durch die weitgehende Überführung der Arbeiten - einschließlich der leistungstragenden Wissenschaftler - in das ISST verfügt das Institut über die erforderliche Fachkompetenz, um Dritten auf diesen Gebieten interessante Leistungen anzubieten.

Das ISST erbringt für öffentliche und private Auftraggeber Forschungs- und Entwicklungsleistungen zu angepaßten Integrations- und Software-Entwicklungsstrategien, zu Methoden, Werkzeugen und Software-Entwicklungsumgebungen. Die Ergebnisse dieser Forschungsarbeiten werden dabei u.a. auf die Analyse und Renovierung existierender Systeme angewandt. Ziel ist die Entwicklung integrierter informationstechnischer Infrastrukturen. Soweit dies inhaltlich und ökonomisch sinnvoll ist, werden Projekte in Zusammenarbeit mit Partnern aus der Softwareindustrie ausgeführt. Für bestimmte Themenbereiche werden dafür strategische Partnerschaften aufgebaut. Dies ist z.B. dort von Bedeutung, wo die Arbeiten bis zur Produktentwicklung führen sollen.

Neben der Durchführung von Forschungsprojekten bietet das ISST

- Software- und Systemnutzern,
- Software- und Systementwicklern und -integratoren,
- Softwaretechnologie-Anbietern,
- Anwendern von Systemen zur werkzeuggestützten Softwareentwicklung,
- privaten Unternehmen und
- öffentlichen Verwaltungen

Unterstützungs- und Beratungsleistungen an.

Für Software- und Systemnutzer, Software- und Systementwickler und -integratoren umfaßt dies

- die Analyse existierender Systeme,
- die Entwicklung und Evaluierung von Integrationsstrategien und -technologien und
- die Migration von Integrationstechnologien in die industrielle Praxis.

Softwaretechnologie-Anbieter werden durch das ISST unterstützt

- bei der Analyse der gegenwärtigen industriellen Software-Entwicklungspraktiken,
- bei der Entwicklung von Technologien zur werkzeuggestützten Softwareentwicklung,
- bei der Migration neuer Technologien in die industrielle Praxis.

Das Angebot an Anwender der werkzeuggestützten Softwareentwicklung beinhaltet

- die Bewertung von kommerziellen Software-Entwicklungswerkzeugen und -Umgebungen unter Berücksichtigung spezifischer Nutzeranforderungen einschließlich individueller Produktberatung,
- die Demonstration derartiger Produkte,
- die Anwendung von Werkzeugen in Pilot- und Demonstrationsprojekten sowie
- die Analyse des Software-Entwicklungsverfahrens im Hinblick auf den reibungsarmen Übergang zu neuen werkzeuggestützten Entwicklungspraktiken.

Privaten Unternehmen und öffentlichen Verwaltungen wird Unterstützung und Beratung geboten

- bei der Beschaffung von integrierten Infrastrukturen und Software für die Qualitätsprüfung von Software und Systemen,

- bei Pilotanwendungen von Beschaffungsprozeduren und Beschaffungshilfsmitteln.

Das ISST ist keiner Technologie und keinem Produkt verpflichtet. Es räumt daher nicht bestimmten Produkten - auch nicht seinen eigenen - einen absoluten Vorrang ein. Dies hat den großen Vorteil, daß es gegenüber allen Partnern als *neutraler unvoreingenommener Berater* agieren kann. Das gilt uneingeschränkt auch dann, wenn das ISST Aufträge Dritter gemeinsam mit Hardwareherstellern, Software- oder Systemhäusern bearbeitet. In solchen Partnerschaften bietet das ISST an, die Rolle des Architekten und Konstrukteurs für neue Systeme und Lösungen zu übernehmen.

Die neutrale Position des ISST ist wesentlich durch den gemeinnützigen Charakter der FhG bedingt, der das ISST als „quasi öffentliche“ Institution ausweist und seine Dienstleistungen auch für die öffentliche Verwaltung besonders interessant macht.

Um potentielle Nutzer mit der Funktion und den Vorteilen neuer Integrationstechniken und neuer Technologien der werkzeuggestützten Softwareentwicklung vertraut zu machen, betreibt das ISST Demonstrations- und Trainingszentren, die kontinuierlich ausgebaut werden. Darüber hinaus gibt es eine enge Zusammenarbeit mit den jetzt überall in der Bundesrepublik (u.a. in Nordrhein- Westfalen, Bayern, Berlin) entstehenden Dienstleistungszentren für die Softwareindustrie.

Die Einführung neuer Technologien im Bereich der Software- und Systemtechnik erfordert vom Fachpersonal eine entsprechende Anpassung seines Wissensstandes. Die gegenwärtige Ausbildung der Software-Spezialisten wird dem zukünftigen Anforderungsprofil nicht gerecht. Aus diesem Grunde sieht das ISST eine wesentliche Aufgabe darin, das Qualifikationsniveau der Softwareentwickler, Systemintegratoren und Manager anzuheben. Dazu bietet das Institut eine Reihe von Schulungs- und Weiterbildungsveranstaltungen an.

1.4 Organisation und wissenschaftliche Bereiche

Das ISST besteht aus den Institutsteilen Berlin und Dortmund. Es ist gegenwärtig in fünf Bereiche mit jeweils einer Anzahl von Arbeitsgebieten gegliedert. Der geplante Ausbau des Instituts wird weitere wissenschaftliche Bereiche hervorbringen.

Die Organigramme des ISST - für Berlin und Dortmund gesondert - mit der Aufteilung in Bereiche und Arbeitsgebiete sind in den Abbildungen 1 und 2 dargestellt.

Der Bereich 1 (GF) existiert separat in Berlin und in Dortmund mit jeweils eigenverantwortlichem Bereichsleiter. Der Bereich ist für die Verwaltung, die Infrastruktur, das Projektcontrolling und die Koordinierung der Akquisition zuständig.

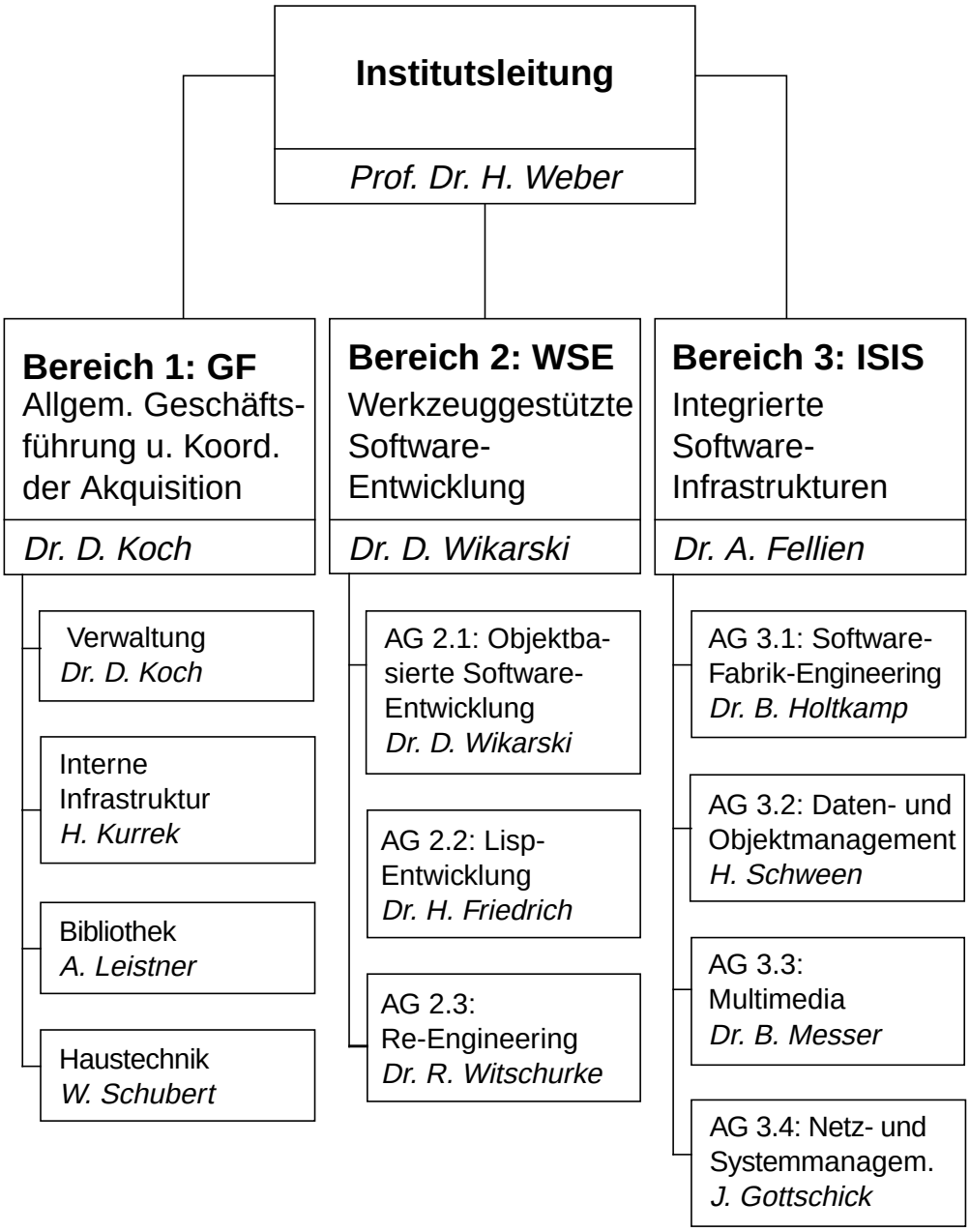


Abb. 1: Organisation Institutsteil Berlin

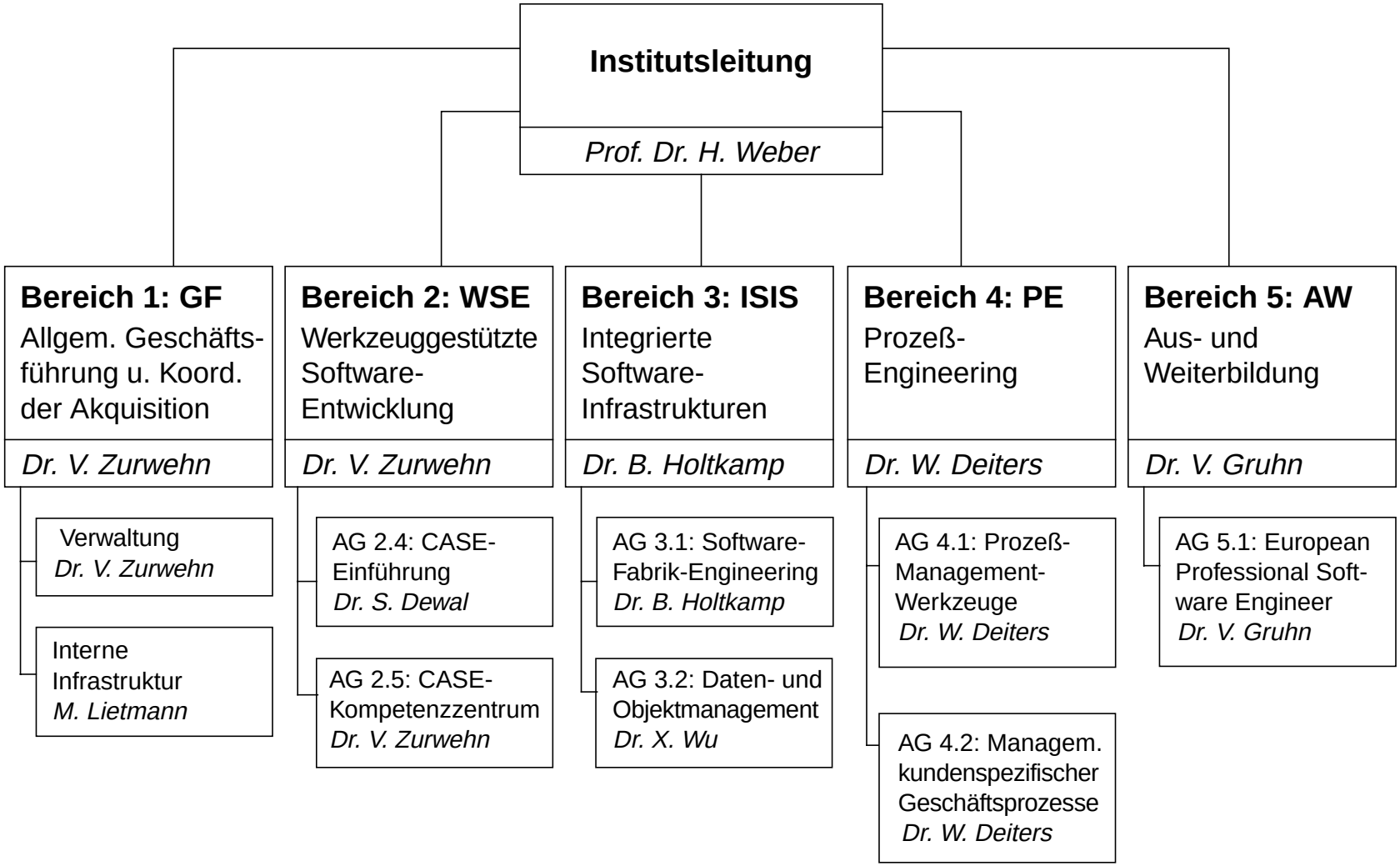


Abb. 2: Organisation Institutsteil Dortmund

Die wissenschaftlichen Bereiche sind teilweise über die beiden Standorte verteilt. Dies soll bewirken, daß die Kompetenz auf wichtigen Themengebieten sowohl in Berlin als auch in Dortmund vorhanden ist. Zur Sicherung der operativen Leitungstätigkeit sind regionale Stellvertreter der Bereichsleiter eingesetzt.

Der Bereich 2 „Werkzeuggestützte Softwareentwicklung“ (WSE) zeichnet verantwortlich für die Arbeitsfelder „Werkzeuge und Umgebungen für die Softwareentwicklung“, „CASE-Einführung“, „Integration und Re-Engineering“ sowie einige allgemeine Aspekte des Arbeitsfeldes „Software Engineering“. Hier gibt es eine enge Beziehung zwischen Berlin und Dortmund vor allem deshalb, weil im Arbeitsgebiet 2.1 in Berlin Arbeiten zur Entwicklung der objektbasierten Spezifikationssprache Π fortgeführt werden, die an der Universität Dortmund 1985 bereits begonnen wurden.

Im Bereich 3 „Integrierte Software-Infrastrukturen“ (ISIS) gibt es eine stark ausgeprägte verteilte Bearbeitung von Themen und Projekten zwischen Berlin und Dortmund. Die durch den Bereich 3 vertretenen Arbeitsfelder sind „Integrierte Software-Infrastrukturen“ und „Datenhaltung und Datenverwaltung in verteilten Infrastrukturen“.

Enge Beziehungen bestehen vor allem zum Bereich 4 „Prozeß-Engineering“ (PE) und zum Bereich WSE im Hinblick auf das Arbeitsgebiet 3.3 (Software-Fabrik-Engineering).

Eine Sonderstellung unter den wissenschaftlichen Bereichen nimmt der Bereich „Aus- und Weiterbildung“ (AW) ein. Unter seine Verantwortung fallen die Vorbereitung und Durchführung von Weiterbildungsveranstaltungen, die Ausarbeitung von Kursen und Lehrmaterialien für alle Kompetenzgebiete des ISST. Dieser Bereich ist für das Institut insoweit von Bedeutung, als die Vermittlung von Wissen und Know-how häufig der erste Schritt zur Akquisition von Projektaufträgen ist.

1.5 Personal und Sachausstattung

Zum Zeitpunkt der Gründung des Instituts (Anfang 1992) waren im ISST 23 wissenschaftliche Mitarbeiter und eine studentische Hilfskraft in Berlin beschäftigt; bis Anfang 1993 stiegen diese Zahlen auf 33 Mitarbeiter und 22 studentische Hilfskräfte in Berlin, sowie auf 15 Mitarbeiter und 10 studentische Hilfskräfte in Dortmund.

Der größte Teil der wissenschaftlichen Mitarbeiter sind Diplom-Informatiker; einige sind Diplom-Physiker, Diplom-Mathematiker oder Diplom-Ingenieure, deren Berufserfahrung sich aber ebenfalls auf das Gebiet der Informatik konzentriert. Zwei Mitarbeiter sind habilitiert, 14 promoviert. Das Durchschnittsalter beträgt 37 Jahre (Berlin) bzw. 32 Jahre (Dortmund). Durch die Verbindungen zu den Universitäten und zu Kooperationspartnern sowie durch interne und externe Fortbildungsveranstaltungen wird eine kontinuierliche Verbesserung der fachlichen Qualifikation der Mitarbeiter erreicht.

Das ISST in Berlin nutzt gemeinsam mit anderen Forschungseinrichtungen der FhG ein Gebäude, wo es über etwa 1000 m^2 Bürofläche verfügt. Die rechentechnische Infrastruktur besteht aus einem Rechnernetz mit leistungsfähigen Servern, etwa 40 UNIX-Maschinen (vor allem SUN-Arbeitsstationen und X-Terminals), einem LISP-Spezialprozessor sowie einigen Macintosh-Rechnern und DOS/Windows-PCs. Die Rechner sind mit moderner Software, darunter leistungsfähigen Werkzeugen und Umgebungen zur Softwareentwicklung, ausgestattet.

Das lokale Rechnernetz des ISST ist in das deutsche Wissenschaftsnetz WIN eingebunden. Die rechentechnische Infrastruktur wird von zwei Mitarbeitern mit Unterstützung studentischer Hilfskräfte betreut. Den Mitarbeitern des ISST steht eine eigene, leistungsfähige Fachbibliothek zur Verfügung.

Da die Dortmunder Außenstelle noch in den Räumen der Universität Dortmund untergebracht ist, nutzt sie zur Zeit deren technische und rechentechnische Infrastruktur mit. Eigene Rechentechnik wird schrittweise beschafft; der wesentliche Aufbau der Infrastruktur erfolgt aber erst mit dem Bezug neuer Räume im Informatik-Centrum Dortmund zum Herbst 1993.

1.6 Haushalt und Finanzierung

Der Betriebsaufwand des ISST betrug 1992 für Berlin und Dortmund zusammen etwa 3,36 Mio. DM. Davon konnten durch Projekterträge mehr als 50% (1,735 Mio. DM) erwirtschaftet werden, der Rest wurde durch die Grundfinanzierung abgedeckt. Die nachfolgende Tabelle zeigt die Zahlen im Detail, aufgeschlüsselt auf die Institutsteile. Auftraggeber für die Auftragsprojekte waren sowohl private Firmen als auch öffentliche Institutionen (s. Abschnitt 2.1).

In Berlin wurden insgesamt etwa 1,4 Mio. DM investiert, davon mehr als 80% in die rechentechnische Infrastruktur. Das Investitionsvolumen in Dortmund betrug 453 TDM.

Für die Renovierung des Institutsgebäudes in Berlin sind 1992 aus zentralen Mitteln der FhG 677 TDM eingesetzt worden.

Tab. 1: Betriebsaufwand und Finanzierung in den Institutsteilen des ISST

ISST	**Berlin in TDM**	**Anteil**	**Dortmund in TDM**	**Anteil**
Betriebsaufwand	2.723		639	
Personalkosten	1.730	63%	482	76%
Betriebsfremde	190	5%	29	4%
Sachaufwand	640	27%	58	9%
Leistungsverrechnung innerhalb der FhG	163	5%	70	11%
Finanzierung	2.723		639	
Projekteinnahmen insgesamt	1.146	41%	589	92%
- öffentliche Projektförderung	954	34%	541	85%
- sonstige Projekterträge	192	7%	48	7%
Grundfinanzierung	1.577	59%	50	8%

2 Fachlicher Ergebnisbericht

2.1 Projektüberblick

Die wissenschaftliche Arbeit in den Bereichen und Arbeitsgebieten ist grundsätzlich in Projekten organisiert, also hinsichtlich der Ziele und Ergebnisse, der eingesetzten Potentiale und des Zeitablaufs überschaubar strukturiert. Im Normalfall gibt es zu einem Arbeitsgebiet mehrere Projekte; die Projektbearbeitung kann dabei über mehrere Bereiche sowie über Berlin und Dortmund verteilt erfolgen.

Der Charakter der Projekte ist unterschiedlich. Zum einen werden (externe) Projekte im Auftrag privater Firmen oder öffentlicher Institutionen bzw. aufgrund öffentlicher Förderung bearbeitet. In diesen Fällen bezahlt der Auftraggeber die Arbeit, nach deren Abschluß ihm ein definiertes Ergebnis zur Verfügung gestellt wird. Zum anderen werden Projekte durch das Institut selbst - sogenannte Eigenforschungsprojekte - aus Mitteln der institutionellen Förderung (Grundfinanzierung) getragen.

Eigenforschungsprojekte haben den Zweck, wissenschaftlichen Vorlauf für die Akquisition neuer Auftragsprojekte zu schaffen und innovative „Produkte“ für spätere Aufträge zu entwickeln. Ein spezieller Aspekt der Eigenforschungsprojekte im ISST ist der interne Technologietransfer zwischen Dortmund und Berlin, um eine einheitliche technische Kompetenz des Gesamtinstituts herzustellen.

Öffentlich geförderte Projekte

Das ISST hat 1992 an drei großen Verbundprojekten mitgearbeitet und mit diesen den größten Teil der Projekterträge erwirtschaftet. Diese Projekte sind:

- das EUREKA-Projekt ESF[1] zur Entwicklung von Konzepten und Werkzeugen für integrierte Software-Infrastrukturen,
- das EUREKA-Projekt ECMA-PCTE zur Implementierung des PCTE-Standards[2] und objektorientierter Erweiterungen,

1. *ESF = EUREKA Software Factory, das größte europäische Forschungsprojekt zum Software Engineering und zu Software-Entwicklungsumgebungen; Beginn 1986*
2. *PCTE = Portable Common Tool Environment*

- das BMFT-Verbundprojekt APPLY[3] zur Entwicklung eines bedarfsgerechten und effizienten LISP-Systems.

ESF wird zur Zeit im Bereich 3, in der nächsten Projektetappe in den Bereichen 2 und 3 (AG „Software-Fabrik-Engineering“) bearbeitet, ECMA-PCTE im Bereich 3 (AG „Daten- und Objektmanagement“) und APPLY im Bereich 2 (AG „LISP-Systeme“).

Auftragsprojekte

An der Finanzierung des ECMA-PCTE-Projekts hat sich die Industrie mit 25% beteiligt. Weitere durch private oder öffentliche Auftraggeber finanzierte - meist kleinere - Projekte waren:

- Konzepte zur Einführung von objektorientierten Techniken (Auftraggeber: SISZ - Software-Industrie Support Zentrum - GmbH, Dortmund);
- MINZE-Studie zu einem Administrationszentrum im Landesamt für Informationstechnik (LIT) (Auftraggeber: LIT Berlin);
- Unterstützung bei der Erstellung des Organisationskonzepts im Projekt FISCUS zur Neukonzeption der EDV-technischen Abwicklung des Steuerwesens (Auftraggeber: Finanzministerien des Bundes und der Länder);
- Analyse des Datenbanksystems „Statice“ im Hinblick auf Möglichkeiten der Effizienzerhöhung (Auftraggeber: Symbolics Systemhaus GmbH);
- Möglichkeiten und Grenzen gemeinsamer Softwareentwicklung in der gesetzlichen Krankenversicherung (Auftraggeber: Landesversicherungsamt NRW);
- Studie zur Erarbeitung einer Realisierungskonzeption für die Informationstechnik in Kreisverwaltungen des Landes Brandenburg (Auftraggeber: Elektronik, Service + Vertrieb GmbH Berlin);
- Analyse softwaretechnologischer Vorgehensweisen und Erstellung von Vorgangsmodellen (Auftraggeber: KSG - Kraftwerks-Simulator Gesellschaft);
- Konzeption einer integrierten EDV
  (Auftraggeber: The Food Professionals).

Die Projekte laufen zum Teil 1993 weiter (FISCUS, KSG) oder münden in Folgeprojekte (LIT).

3. APPLY = A Practical and Portable LISP „Environment“; „apply“ ist der Name einer für LISP zentralen Funktion.

„Produkt“-Entwicklungs-Projekte

Der Begriff „Produkt“ ist hier in einem allgemeineren Sinne zu verstehen, der z.B. auch ein Paket von Projektdienstleistungen mit einer eindeutigen Identifikation für den Markt umfaßt. Mit solchen Produktentwicklungen muß das ISST einen gewissen Markt erreichen, die „Produkte“ auf dem Markt mehrfach absetzen und so eine längerfristige Finanzierung sichern.

Im Jahr 1992 in „Produkt“-Entwicklungsprojekten erarbeitete oder in Entwicklung befindliche „Produkte“ sind

- das European Professional Software Engineering Program (EPSE)
  In einem europäischen Verbund ist ein Ausbildungsprogramm zum Software-Ingenieur entwickelt worden, das sich insbesondere an industrielle Softwareentwickler wendet (s. Abschnitt Aus- und Weiterbildung).
- ein CASE Deployment Kit
  Dieses faßt Konzepte, Methoden und Werkzeuge zur Vorbereitung und Realisierung der Einführung der werkzeuggestützten Softwareentwicklung in Unternehmen zusammen; erste Teile sind realisiert (s. Abschnitt CASE-Einführung).
- ein Re-Engineering Kit
  Das Re-Engineering-Kit soll die methodische und Werkzeugbasis für die Durchführung konkreter Re-Engineering-Projekte liefern (s. Abschnitt Re-Engineering).
- ein Vorgangsmanagement Kit
  (s. Darstellung zum Bereich Prozeß-Engineering).

Interne Technologietransfer-Projekte

Eigenforschungsprojekte, die auch einen Technologietransfer zwischen Dortmund und Berlin zum Ziel hatten, waren 1992

- Π-Konzepte (Software-Modularität)
  Hier ging es insbesondere um die abschließende Definition der objektbasierten Spezifikationssprache Π und die Schaffung einer dazugehörigen Entwicklungsumgebung.
- Software-Infrastrukturen
  Dieses Projekt diente der Verbreitung und Festigung des Know-how auf dem für das ISST maßgebenden Gebiet „Software-Infrastrukturen“ sowie der Vorbereitung von Kundenprojekten (LIT, BERKOM, Diensteplattform; s. Abschnitt „Integrierte Software-Infrastrukturen“).
- Datenhaltung und Datenverwaltung
  Das Projekt hatte die Aufgabe, die Überführung des Projekts ECMA-PCTE nach Berlin vorzubereiten und notwendige Begleitforschung zu sichern.

Die Inhalte einiger Projekte werden im einzelnen in den nachfolgenden Abschnitten beschrieben.

2.2 Werkzeuggestützte Softwareentwicklung

2.2.1 Zielsetzung des Bereichs

Der Stau bei der Entwicklung von Anwendungssoftware erfordert, daß Softwareentwicklung in Zukunft weitestgehend werkzeuggestützt erfolgt. Daher ist ein Hauptarbeitsgebiet des Bereichs „Werkzeuggestützte Softwareentwicklung“ die Entwicklung leistungsfähiger Software-Entwicklungsmethoden und -werkzeuge.

Ein strategisches Ziel des ISST besteht darin, für ein breites Spektrum von software- und systemtechnischen Anwendungen integrierende Methoden und weitgehend hardwareunabhängige Werkzeuge und Plattformen bereitzustellen, die eine *ingenieurmäßige Entwicklung wiederverwendbarer Softwarebausteine* ermöglichen. Die verwendeten Methoden und Werkzeuge sollen dabei gewährleisten, daß die erstellte Software

- nachweisbar bestimmten Qualitätsanforderungen genügt,
- noch im Entwicklungsprozeß im Hinblick auf ihre Funktionalität erprobt werden kann (Prototyping) und
- eine möglichst unproblematische Portierung auch auf parallele und verteilte Hardware gestattet.

Darüber hinaus soll durch Werkzeuge und Methoden eine hohe Effizienz der Softwareentwicklung ermöglicht werden.

Die Entwicklung wiederverwendbarer Softwarebausteine zielt auf eine *geplante* Wiederverwendung. Daneben ist eine *ungeplante* Wiederverwendung existierender Systeme und Systemkomponenten - z.B. auf anderen Plattformen und für veränderte Anforderungen - von wesentlicher Bedeutung (Re-Engineering). Dies erfordert z.B. Methoden und Werkzeuge für den Umkehrungsprozeß (Reverse Engineering) zur eigentlichen Softwareentwicklung (Forward Engineering).

In der werkzeuggestützten Softwareentwicklung verfolgt das ISST das Ziel, auf der Basis eines Kerns fundamentaler Konzepte und Sprachen ingenieurtechnische Methoden des Software Engineering zu entwickeln und für diese Methoden adäquate Werkzeuge bereitzustellen.

Die Verfügbarkeit leistungsfähiger Software-Entwicklungsmethoden und -werkzeuge allein ist aber nicht ausreichend; dies illustriert die heutige Situation bei der Anwendung von CASE-Systemen. Eine ganze Reihe leistungsfähiger CASE-Systeme ist bereits auf dem Markt verfügbar, der bisher durch CASE und CASE-Werkzeuge erreichte Nutzen ist allerdings eher gering. Dies hat verschiedene Ursachen. Zum einen weisen die verfügbaren Werkzeuge eine Reihe von Mängeln auf, zum anderen fehlen wichtige Voraussetzungen, um derartige Werkzeuge in der Praxis auch wirksam einzusetzen. Damit die Einführung von Software-Entwicklungswerkzeugen und -umgebungen in die industrielle Praxis gelingt, müssen die dabei notwendigen Migrationsprozesse analysiert, praktikable Konzepte für die Einführung definiert und Unternehmen bei ihrem Strukturwandel unterstützt werden.

2.2.2 Arbeitsgebiete

Die beschriebene Zielsetzung wird im Bereich „Werkzeuggestützte Softwareentwicklung“ auf verschiedenen Arbeitsgebieten verfolgt:

- **Π-Konzepte (Software-Modularität)**
  Dieses Arbeitsgebiet umfaßt die Ausarbeitung und Weiterentwicklung von Basistechniken für die Entwicklung modularer Softwaresysteme. Zentraler Gegenstand der Arbeiten ist die objektbasierte Spezifikationssprache Π. Der Schwerpunkt lag 1992 neben der Weiterentwicklung der Sprache Π zu einer anwendungsfähigen Version auf deren Implementation und der Fertigstellung des Prototyps einer entsprechenden Entwicklungsumgebung PISA (PI Software Engineering Assistant).
- **Entwicklung von LISP-Systemen**
  Inhalt der Arbeiten ist die Schaffung von bedarfsgerechten und effizienten LISP-Systemen (APPLY-Projekt). Ohne auf die LISP auszeichnende hohe Flexibilität zu verzichten, soll es möglich werden, LISP-Anwendungen problemlos in die übrige Softwarewelt zu integrieren (z.B. gegenseitige Aufrufbarkeit von LISP- und C-Programmen) und dabei eine Speicher- und Laufzeiteffizienz zu erreichen, die der von C-Programmen vergleichbar ist.
- **Re-Engineering**
  Re-Engineering bedeutet, Softwaresysteme rechnergestützt an neue Anforderungen anzupassen bzw. sie bedarfsgerecht qualitativ zu verbessern. Wichtige Teilaspekte sind die Analyse von Software, Verfahren des Reverse Engineering und die Umstrukturierung von Softwaresystemen. Das Re-Engineering wird im ISST vor allem unter einem pragmatischen, anwendungsbezogenen Gesichtspunkt behandelt. So besteht das wesentliche Ziel in der Entwicklung eines Re-Engineering Kit, das die Durchführung von Projekten zum Re-Engineering konkreter Softwaresysteme unterstützen soll.
- **CASE-Einführung**
  Ziel der Arbeiten ist die Entwicklung eines Bündels von Konzepten, Methoden und Werkzeugen, das die Einführung der werkzeuggestützten Softwareentwicklung in die industrielle Praxis unterstützt (CASE Deployment Kit). Dabei müssen

die organisatorischen, technischen und personellen Voraussetzungen des jeweiligen Unternehmens berücksichtigt werden. Eine Analyse des status quo der Softwareentwicklung und des Aufgabenprofils des Unternehmens kann dazu beitragen, die Auswahl von Werkzeugen zu bestimmen und das Unternehmensmodell zu verbessern.

- **CASE-Kompetenzzentrum**
  Ein CASE-Kompetenzzentrum soll interessierten Anwendern die Möglichkeit geben, Software-Entwicklungswerkzeuge und -umgebungen verschiedener Hersteller im direkten Vergleich betrachten und beurteilen zu können. Die Bewertungsergebnisse des ISST sollen in einer Werkzeug-Datenbank zusammengefaßt und den Kunden zur Verfügung gestellt werden.

2.2.3 Speziationssprache Π und Π-Umgebung PISA

Ziele der Π-Entwicklungen

Die objektbasierte Spezifikations- und Implementationssprache Π wurde mit dem Ziel entwickelt, eine systematische Produktion großer, nebenläufiger und verteilter Softwaresysteme zu ermöglichen. Dabei soll durch die Sprache und die eingesetzten Entwicklungswerkzeuge ein rigoroser softwaretechnischer Stil erzwungen werden, der sich durch eine strenge Modularisierungsdisziplin auszeichnet und eine gute Kontrolle von Seiteneffekten ermöglicht.

Der hohe Formalisierungsgrad der Schnittstellen von Π erlaubt es, elementare Bausteine bzw. Module wiederzuverwenden, wodurch ein großes Maß an Sicherheit in der Funktionalität der erstellten Programme erreicht werden kann. Die Wiederverwendbarkeit der Module bezieht sich hier in erster Linie auf neu zu implementierende Π-Module, ist aber durch die Anwendung von Re-Spezifikation auch für bereits vorhandene Softwarebausteine in anderen Sprachen möglich.

Die Anwendung von Π zum Entwurf und zur Wiederverwendung von Software soll prinzipiell unter Nutzung von Softwarewerkzeugen erfolgen. Neben der Unterstützung der interaktiven Erstellung von Quelltexten sollen diese auch die Ausführung unvollständiger Spezifikationen und die Überprüfung von Konsistenz- und Kontextbedingungen gestatten und damit die Korrektheit der Spezifikationen und der daraus abgeleiteten Programme sichern helfen.

Ein Anwendungsgebiet der Sprache Π ist ihr prototypischer Einsatz als Komponentenbeschreibungssprache in Software-Fabriken. Entsprechende Arbeiten sind im ESF-Projekt vorgesehen.

Die Sprache Π und ihre Werkzeuge sollen eine wesentliche Rolle bei der Herausbildung einer weitgehend homogenen Methoden- und Werkzeug-Basis spielen. Sie ist

damit im ISST als ein projektübergreifender Integrationsfaktor anzusehen und bildet einen Teil des Kerns fundamentaler Konzepte und Sprachen zur Entwicklung tragfähiger Methoden des Software Engineering.

Vorgehensweise

Das prinzipielle Vorgehen sowohl bei der Strukturierung des Software-Entwicklungsprozesses unter Anwendung von Π als auch beim Entwurf der Sprache selbst beruht auf zwei fundamentalen Prinzipien der Softwareentwicklung: „Teile und herrsche“ („divide and conquer“) und „Separierung von Charakteristiken“ („separation of concerns“).

Das erste Prinzip, das dem Abstraktionsprinzip „Verbergen von Information“ („information hiding“, auch: Geheimnisprinzip) entspricht, wird in der Sprache Π durch eine hierarchische Struktur der Softwarebeschreibung realisiert: Jedes zu spezifizierende Softwaresystem wird als Konfiguration von Komponenten dargestellt. Dabei können die Komponenten nebenläufig ausführbare Module („concurrently executable modules“, abgekürzt CEM) oder selbst wieder Konfigurationen von Komponenten sein. Die CEMs sind nach einem einheitlichen Prinzip strukturiert. Nach Auflösung der beschriebenen rekursiven Beziehung bilden sie die Basiskomponenten eines jeden mit Π spezifizierten Softwaresystems.

Dem zweitgenannten Prinzip der Softwareentwicklung entspricht das Abstraktionsprinzip „Weglassen von Information“ („information neglection“). Es wird in der Sprache Π durch verschiedene Sichten („views“) realisiert, aus denen jeder Modul spezifiziert werden kann. Grundlegend sind dabei die Typ-Sicht (Type View), die imperative Sicht (Imperative View) und die Nebenläufigkeitssicht (Concurrency View), in denen die Schnittstellen eines Moduls sowie seine interne Struktur jeweils durch algebraische Spezifikation, durch die Angabe imperativer Programme für die Operationen des Moduls sowie durch die Einschränkung ihrer möglichen Nebenläufigkeit durch Pfadausdrücke definiert werden. Die Offenheit des Sichtenkonzepts ermöglicht es, entsprechend dem Bedarf von Anwendungen und dem Stand der Forschung weitere Sichten, z.B. über zeitliche Einschränkungen des Programmverhaltens (Performance view) oder über die Zuordnung von Softwarekomponenten zu Prozessoren (Distribution view), zu integrieren. So können existierende Spezifikationen und Werkzeugkomponenten für die bestehenden Sichten gültig bleiben.

Die detaillierte Ausarbeitung eines Vorgehensmodells zur Entwicklung von Softwaresystemen mit Π ist noch Aufgabe der Forschung. Sie erfolgt unter anderem anhand der prototypischen Anwendung der Sprache und ihrer Werkzeuge am ISST sowie in Projekten in Forschung und Lehre. Ein wesentliches Merkmal des Vorgehensmodells ist die Integration der Werkzeuge in ein unterlegtes Prozeßmodell für die inkrementelle Entwicklung von Spezifikationen.

Ergebnisse

- **Syntax Π92**
  Auf der Grundlage der vom Lehrstuhl für Softwaretechnologie der Universität Dortmund bereitgestellten Version der Sprache Π wurde eine modifizierte Syntaxversion (Π92) erarbeitet. Diese bildet für absehbare Zeit die Basis zur Spezifikation mit Π und für die Implementation der Software-Entwicklungswerkzeuge für Π.

- **Methodik und Werkzeugkonzepte für die Anwendung von Π**
  Für die erfolgreiche Anwendung von Π ist eine wesentliche Voraussetzung, daß eine geeignete Methodik für Entwurf, Spezifikation und Implementierung sowie entsprechende Softwarewerkzeuge vorhanden sind.
  Im Rahmen einer einheitlichen Programmierstrategie für das ISST wurde ein Entwurf für Π-konforme Programmier-und Dokumentationsrichtlinien erstellt.
  Aufbauend auf dem an der Universität Dortmund entwickelten Prototypen eines syntaxgesteuerten Editors PILS wurden Konzepte für die Architektur einer Π-Entwicklungsumgebung und ihrer Komponenten CEM-Editor, Konfigurationseditor (GRECO), Bibliotheksverwalter, Compiler und Kontextchecker erarbeitet. Darüber hinaus wurden Untersuchungen zu den Teilkomponenten dieser Umgebung Π-Objektmanagementsystem (POMMES), Fensterhierarchie, Fremdtextbehandlung, Fehlerbehandlung und Filesystemanbindung durchgeführt.

- **PI Software Engineering Assistant (PISA)**
  Auf der Grundlage dieser Konzepte wurde ein erster Prototyp der PISA-Architektur implementiert. Anstelle eines in spätere Versionen zu integrierenden Objektverwaltungssystems wurde zunächst eine direkte Anbindung der Umgebung an das UNIX-Filesystem vorgenommen.

- **Konzepte zur Behandlung von Π-Spezifikationen**
  Wesentliche Funktionen der Entwicklungsumgebung PISA bestehen darin, die kontextsensitive Korrektheit von Π-Spezifikationen zu überprüfen, um daraus anschließend ausführbare C-Programme zu generieren.
  Da jeder Π-Modul (CEM) gleichzeitig aus verschiedenen Sichten spezifiziert sein kann, erfordert die Korrektheitsanalyse ein bestimmtes Herangehen. Das erarbeitete Konzept zur Überprüfung der Konsistenz und kontextsensitiven Korrektheit von Spezifikationen sieht daher unter anderem die Generierung redundanter Teile von Spezifikationen (beabsichtigte Redundanz im Hinblick auf verschiedene Sichten) vor.
  Um eine möglichst vielseitige Anwendbarkeit der Werkzeuge im Hinblick auf die Ausführung von Spezifikationen zu erreichen, wurde ein Konzept zur Compilierung der Typ-Sicht entwickelt. Das Konzept ist auf eine wesentliche Teilklasse der in dieser Sicht möglichen Spezifikationen (sogenannte konstruktive Spezifikationen) anwendbar.

- **Dokumentation**
  Sowohl für die Akzeptanz der Sprache und der Werkzeuge als auch für deren spätere Wartung und Modifizierbarkeit ist eine gute Dokumentation von entscheidender Bedeutung. Daher wurde diesem Aspekt große Aufmerksamkeit geschenkt.
  Die Software wurde adäquat kommentiert und ein mit instruktiven Beispielen ausgestattetes Programmierhandbuch für Π sowie ein PISA-Nutzerhandbuch wurden erstellt. Die Konzepte für Sprache und Werkzeuge wurden in internen und publizierten Forschungsberichten dargestellt.

Veröffentlichungen

[ABF+92]; [Ada92]; [Bud92a]; [Bud92b]; [Bud92c]; [Bud92d]; [Gab92a]; [Gab92b]

2.2.4 Entwicklung von LISP-Systemen

Ziele

Die Programmiersprache LISP ist eine der ältesten Programmiersprachen. Sie besitzt als dynamisch getypte Sprache gegenüber den üblicherweise in der Softwareentwicklung verwendeten Programmiersprachen den Vorteil einer höheren Flexibilität. In der Künstlichen Intelligenz (KI) ist sie die am meisten verbreitete Programmiersprache und wird für die Implementation komplexer KI-Systeme (z.B. Systeme zum Sprachverstehen, zum Bildverstehen, zum Theorembeweisen u.a) verwendet. Sie eignet sich gut zum Prototyping und zum sogenannten explorativen Programmieren. Außerhalb der Künstlichen Intelligenz kam sie bisher relativ selten zum Einsatz (z.B. in den weit verbreiteten Systemen AUTOCAD und EMACS, gegenwärtig auch als Systemprogrammiersprache der neuen Computergeneration der Firma Apple). Dies ist im wesentlichen darauf zurückzuführen, daß LISP-Programme meist deutliche Nachteile in bezug auf Laufzeit und Speicherplatzbedarf haben und die Kopplung mit Programmen anderer Programmiersprachen Probleme bereitet.

Dies ist der Ausgangspunkt für das BMFT-Verbundprojekt APPLY, in dem das ISST gemeinsam mit der Universität Kiel und der GMD Sankt Augustin an diesen Problemen arbeitet.

Ziel des APPLY-Projektes ist es, eine Strategie zur LISP-Implementation zu entwikkeln und ein entsprechendes LISP-System zu realisieren, so daß

- LISP-Anwenderprogramme eine Speicherplatz- und Laufzeiteffizienz haben, die der von äquivalenten C-Programmen vergleichbar ist und
- eine natürliche Integration von LISP-Programmen in die übrige Softwarewelt möglich ist, ohne daß die Vorteile von LISP verloren gehen.

LISP soll dadurch die notwendige Akzeptanz bei Softwareproduzenten und Anwendern finden und in der Nutzung nicht nur auf die KI-Forschung beschränkt bleiben. Es soll die Lücke geschlossen werden, die sich zwischen dem Stand der KI-Forschung und der verfügbaren KI-Basissoftware aufgetan hat, weil die Effizienznachteile der KI-Basissoftware die rasche kommerzielle Nutzung der Forschungsergebnisse der KI zunehmend behindern oder gar unmöglich machen.

Im Rahmen von APPLY erfolgt eine Koordinierung der an verschiedenen deutschen Forschungsinstituten und Hochschulen laufenden Arbeiten zur Weiterentwicklung von LISP. Durch die Konzentration der Erfahrungen soll der Einfluß auf die internationale Standardisierung von LISP verbessert werden. Die Ergebnisse des Sprachdesigns und die Implementationserfahrungen sollen insbesondere in die europäischen Standardisierungsbemühungen (EuLisp) eingebracht werden.

Vorgehensweise

Die Implementationsstrategie[4] existierender LISP-Systeme ist nicht geeignet, die oben genannten Forderungen zu erfüllen.

Deshalb wird eine radikal neue Strategie bei der Implementation von LISP entwickelt. Diese unterscheidet sich von der klassischen Herangehensweise vor allem in den folgenden Punkten:

- **Klare Trennung von Entwicklung und Anwendung**
  Fertige Programme benötigen die Flexibilität und den Leistungsumfang von LISP nicht in dem Maße, wie es während der Programmentwicklung notwendig ist. Anwendungsprogramme können durch Eliminierung des nicht benötigten Leistungsumfangs nicht nur bedeutend kleiner, sondern auch effizienter werden. Entsprechende Optimierungen für die Applikation können sowohl die Verkleinerung des Funktionssatzes als auch die Anpassung von Objektrepräsentation und Speicherverwaltung umfassen. Sie sind jedoch nur auf der Grundlage einer intensiven Programmanalyse möglich.

---

4. Die klassische Implementationsstrategie für LISP ist durch folgende Merkmale gekennzeichnet:

- keine Trennung zwischen Entwicklung(ssystem) und Anwendung(ssystem); es ist immer das vollständige System aktiv, unabhängig davon, ob der gesamte Leistungsumfang benötigt wird oder nicht.

- die Integrierbarkeit in andere Softwareprodukte ist nicht gegeben; LISP ist ein abgeschlossenes System;

- LISP-Systeme sind in zentralen Teilen nicht konfigurierbar;

- Compiler können nicht genügend Informationen aus den Programmen gewinnen, um eine mit C vergleichbare Effizienz zu erzielen.

- **Von LISP unabhängige Applikationen**
  Im Gegensatz zur klassischen Vorgehensweise bei LISP ist APPLY auf die Compilierung von Anwendungsprogrammen ausgerichtet. Das Compilationsergebnis sollen übliche Objektfiles oder C-Programme sein, die mit Betriebssystemmitteln zu fertigen Programmen verbunden werden können. Dies ist eine wesentliche Voraussetzung für die wirkliche Integration in die Softwarewelt und für einen hohen Grad der Wiederverwendbarkeit von LISP-Programmen. Der Interpreter ist nur noch Bestandteil der Entwicklungsumgebung, zu der unter anderem auch ein inkrementeller Compiler und die Modulverwaltung gehören.

- **Globale Programmoptimierung**
  Dem Compiler müssen zur Erzeugung effizienter Applikationen ausreichend Informationen zur Verfügung stehen, die nur durch globale Programmanalyse ableitbar sind. Die zentrale Lösungsidee besteht darin, auf der Basis eines geeigneten Modulkonzepts die Compilierungseinheiten so zu vergrößern, daß eine globale Programmanalyse mit einer darauf aufbauenden Optimierung möglich wird. Bei der Erzeugung von Applikationen braucht nur noch die Semantik der Module oder der vollständigen Programme erhalten zu bleiben, nicht aber die jeder einzelnen Funktion. Aus der Semantik der Systemfunktionen und Deklarationen müssen Informationen zu Datentypen (Typinferenz), zu Seiteneffekten, zur Lebensdauer, zu gemeinsamen Teilausdrücken, zur Menge der auf den Daten angewandten Operationen, zur Struktur und Verwendung von Listen etc. gewonnen werden. Diese Informationen sollen den Compiler in die Lage versetzen,

  - Programmtransformationen durchzuführen,
  - die dynamische Typprüfung zu reduzieren,
  - effiziente Datenrepräsentationen zu verwenden und
  - den Aufwand für die Speicherverwaltung zu minimieren.

  Weiterhin muß der Compiler die (implizite) Nutzung effizienter Teilsprachen (Beispiel: multiple values treten nicht auf) erkennen können, um daraus entsprechende Optimierungen abzuleiten und eine der jeweiligen Applikation angepaßte Laufzeitumgebung bereitzustellen.

- **Konfigurierbarkeit**
  Um eine hohe Effizienz zu erzielen, müssen zentrale Teile des LISP-Systems an gegebene Hardware- und Betriebssystembedingungen angepaßt werden. Eine einfache Portabilität reicht nicht aus, um systemspezifische Eigenschaften zur Effizienzsteigerung ausnutzen zu können. Deshalb muß der Kern des LISP-Systems modular und mit klaren Schnittstellen aufgebaut sein, so daß zentrale Teile, wie Speicherverwaltung und Objektrepräsentation, austauschbar sind. Auch eine Anpassung des LISP-Kerns an eine bestimmte Applikation kann eine Effizienzsteigerung bewirken. Dabei kann ausgenutzt werden, daß viele Applikationen nicht den gesamten Leistungsumfang von LISP benötigen. In diesen Fällen können bestimmte zentrale Teile der Laufzeitumgebung durch effizientere, bereits vorgefertigte Realisierungen ersetzt werden.

- **Verbessertes Sprachdesign**
  Eine wesentliche Voraussetzung für eine hohe Effizienz ist ein auf dieses Ziel ausgerichtetes Sprachdesign. Dieses muß sich durch bessere Möglichkeiten zur Programmspezifikation, durch ein Modulbeschreibungskonzept, ein verbessertes Typkonzept, eine klare Gliederung und die Möglichkeit zur Teilsprachendefinition auszeichnen. Zur Erfüllung dieser Forderungen bietet EuLisp gute Voraussetzungen. EuLisp hat zudem den Vorteil, daß auf die Definition durch das ISST noch Einfluß genommen werden kann und wird.
  Aus diesen Gründen baut APPLY auf der aktuellen EuLisp-Definition auf. Eine hinreichende Kompatibilität zu COMMON LISP wird durch die Definition eines COMMON LISP-Subsets und dessen Realisierung in einem COMMON LISP-Modul erreicht.

Ergebnisse

Der Schwerpunkt der Arbeiten zum APPLY-Projekt am ISST lag 1992 auf den Arbeitspaketen „Implementationstechnologie“, „Definition der Zwischensprache“, „Transformatoren auf die Zwischensprache“, „Compilierung in Maschinencode“ und „Typen und Typinferenz“. Zu diesen Arbeitspaketen liegen Implementationsergebnisse vor. Weitere Aktivitäten betrafen die Arbeitspakete „Speicherverwaltung und Objektrepräsentation“ sowie „Interface zu C“ und die Mitarbeit an der internationalen LISP-Standardisierung.

Die Ergebnisse zu den Arbeitspaketen sind im einzelnen:

- **Implementationstechnologie**
  Es wurde die Sprache TAIL und damit eine Technologie zur effizienten Implementation von LISP in LISP entwickelt. Um den entwickelten Compiler möglichst einfach mit sich selbst compilieren zu können, wurde ein EuLisp-Kompatibilitätsmodul erstellt.
- **Definition der Zwischensprache**
  In enger Zusammenarbeit mit der Universität Kiel wurde die LISP-nahe Zwischensprache LZS entwickelt. Eine maschinennahe Zwischensprache MZS wurde definiert und implementiert. Zur Erzeugung von optimierten Assemblerprogrammen ist eine Abbildung auf diese Zwischensprache (MZS) notwendig.
- **Compilierung in Maschinencode**
  Auf der Basis einer Beschreibung für Prozessoren und Assembler (Maschinenbeschreibung) wurde eine erste Version eines Codegenerators für den SPARC-Prozessor entwickelt.
- **Typen und Typinferenz**
  Die Laufzeiteffizienz von Applikationen der (dynamisch getypten) Programmiersprache LISP kann durch statische Inferenz von Datentypen entscheidend verbessert werden.

Die Analyse zur Ableitung der Datentypen erfolgt in zwei Schritten:

1. Typinferenz durch die Analyse des Rumpfes jeder einzelnen Funktion;
2. Abgleich zwischen den Funktionen und ihren Aufrufen (Applikationen).

Das implementierte Typinferenzsystem erlaubt durch die Verwendung von verfeinerten Typverbänden und generischen Typschemata, Argument- und Ergebnistypen polymorpher Funktionen präzise zu beschreiben. Mit Hilfe vordefinierter Typschemata der Standard-Funktionen und -Konstanten sowie bereits abgeleiteter Typschemata benutzerdefinierter Funktionen können neue Typschemata ermittelt werden. Zur Inferenz wird ein abgewandeltes Unifikationsverfahren verwendet.

Das entwickelte Typinferenzverfahren wurde implementiert.

Veröffentlichungen

[BCF+92a]; [BCF+92b]; [BCF+92c]; [BCF+92d]; [FHK+92a]; [FHK+92b]; [FR92]; [HR92]; [KK92]; [KM92]; [Kri92a]; [Kri92b]; [Kri92c]

2.2.5 Re-Engineering

Ziele

Die Ausarbeitung von Methoden für die Sanierung bereits existierender Software mit Hilfe des Re-Engineering ist eine wichtige Aufgabe des ISST.

Das Re-Engineering gilt als der Sektor der Softwarewirtschaft mit den zukünftig höchsten Auftragsraten, weil der Bedarf besteht, die Investitionen der letzten 30 Jahre trotz Alterung der Software nach Möglichkeit zu erhalten. Die Forschung hat dem Re-Engineering bisher nur geringe Aufmerksamkeit gewidmet; ungeachtet dessen gibt es bereits eine Reihe pragmatischer Ansätze und Werkzeuge zur Lösung praktischer Probleme.

Für das ISST ergibt sich hier ein interessantes Feld für die anwendungsorientierte Forschung und die Durchführung konkreter Anwendungsprojekte.

Anliegen des Re-Engineering ganz allgemein ist die Qualitätsverbesserung existierender Softwaresysteme. Dabei bedeutet hohe Software-Qualität eine möglichst gute Befriedigung der Nutzerwünsche. Re-Engineering-Verfahren (speziell: Reverse-Engineering für Nachspezifikation, Nachdokumentation u.a.) spielen aber auch eine Rolle bei der Herstellung neuer Software.

Die Arbeiten zum Re-Engineering im ISST zielen darauf ab, ein Bündel von Methoden, Prozeduren und Werkzeugen (Re-Engineering Kit) zur Verfügung zu haben, um

konkrete Projekte zum Re-Engineering von Softwaresystemen durchführen zu können. Damit ergeben sich zunächst die folgenden Ziele:

- Entwicklung eines Re-Engineering-Modells,
- Entwicklung eines Evaluierungs- und Klassifikationskonzepts für Re-Engineering-Werkzeuge,
- Entwicklung von Re-Engineering-Methoden unter Berücksichtigung der Verwendung existierender Werkzeuge,
- Durchführung von Pilotprojekten,
- Ausarbeitung von Lehrmaterialien.

Vorgehensweise

Grundlage der Entwicklung eines Re-Engineering-Modells ist die detaillierte ebenenorientierte Analyse von Software.

Folgende Softwareebenen werden unterschieden:

Nutzerebene:	Anwendungssystem (Anforderungen);
Systemebene:	Softwaresystem, Datenhaltung, Betriebsarchitektur;
Komponentenebene:	Softwarekomponenten;
Programmebene:	Programme, Module, Steueranweisungen, Segmente;
Codeebene:	ausführbarer Code.

Die auf den verschiedenen Ebenen einzusetzenden Methoden sind sehr unterschiedlich.

Für die Programmebene kann auf eine ganze Reihe kommerzieller CARE-Werkzeuge[5] zurückgegriffen werden. Verschiedene spezielle CARE-Werkzeuge und Software-Entwicklungsumgebungen mit Re-Komponente werden auf ihre Eignung (und Kombinationsfähigkeit) hin untersucht. Eine Teilmenge wird für die Arbeiten zum praktischen Re-Engineering ausgewählt. In Übereinstimmung mit dem Ziel, im ISST eine homogene Methoden- und Werkzeugbasis zu schaffen (s. dazu die Ausführungen zu Π), sieht die Konzeption vor, daß die bei der Analyse von Programmen erzeugten Spezifikationsdaten in die Spezifikationssprache Π transformiert und dann vervollständigt werden.

5. CARE = Computer Aided Re(verse) Engineering

Aufgaben beim Re-Engineering sind z.B.:

- Bestandsaufnahme (Reverse-Engineering: Nachdokumentation, Nachspezifikation),
- Restrukturierung mit bzw. ohne Veränderung der Funktionalität,
- Re-Engineering von Daten und Datenbanken.

Weniger untersuchte, aber in ihren Auswirkungen nicht minder wichtige Bereiche, sind auf der Systemebene:

- Re-Engineering der Administration,
- Analyse der Ressourcen,
- Re-Engineering der Hilfesysteme und Textunterlagen.

Die Arbeiten im ISST konzentrieren sich auf die Systemebene, für die es bisher nur wenige (Teil-)Lösungen gibt.

Ergebnisse

Die bisher erzielten Ergebnisse sind:

- Konzeption für ein allgemeines Modell für das Re-Engineering von Softwaresystemen,
- Vorschlag für ein pragmatisches Modell für die Analyse und Integration bestehender Softwarekomponenten,
- Untersuchung ausgewählter Werkzeuge für das Re-Engineering,
- Ausarbeitung von Konzept und Material eines Qualifizierungsprogramms für Informatiker auf diesem Gebiet.

Veröffentlichungen

[Wit92]; [Web92r]

2.2.6 CASE-Einführung

Aufgabenstellung

Die Erfahrung der vergangenen Jahre hat gezeigt, daß die Einführung von CASE-Methoden und -Werkzeugen in den Software-Entwicklungsabteilungen der Unternehmen vielfach nicht den erhofften Effekt gebracht hat. Eine wesentliche Ursache dafür ist, daß der enge Zusammenhang zwischen der Art der zu entwickelnden Software, den allgemeinen betrieblichen Abläufen im Unternehmen, dem bei der Softwareentwicklung praktizierten Vorgehen, den verwendeten Methoden und den benutzten Werkzeugen häufig nicht genügend beachtet worden ist.

An dieser Stelle setzen die Arbeiten des ISST zur CASE-Einführung an. Ziel der Entwicklungen ist ein „CASE Deployment Kit“, das Richtlinien, Analyse- und Entscheidungshilfsmittel sowie Werkzeuge enthält. Es soll die Bestimmung des jeweils sinnvollen Vorgehens bei der Einführung der werkzeuggestützten Softwareentwicklung, die Auswahl der adäquaten Methoden und Werkzeuge sowie die Festlegung eines geeigneten Unternehmensmodells unterstützen.

Zu beachten ist dabei, daß ein geordneter Übergang vom ursprünglichen zu dem neuen, vom Unternehmensmodell beschriebenen Vorgehen ermöglicht wird. Dabei soll das Modell die technischen und personellen Möglichkeiten für den Einsatz von Methoden und Werkzeugen angemessen berücksichtigen.

Vorgehensweise

Für den Migrationsprozeß bei der Einführung der werkzeuggestützten Softwareentwicklung wird eine Vorgehensweise vorgeschlagen, die durch die folgenden Schritte bestimmt ist:

1. Beschreibung des zu beratenden Unternehmens
   Ausgangspunkt für die Einführung von CASE-Technologien in ein Unternehmen ist die Beschreibung der organisatorischen, technischen und personellen Voraussetzungen. In einem mehrtägigen Workshop wird mit Hilfe der Mitarbeiter ein Unternehmensmodell erstellt, das die betrieblichen Abläufe, die eingesetzten Methoden, Software- und Hardwareplattformen und Werkzeuge sowie die Softwaretechnologischen Kenntnisse der Mitarbeiter beschreibt.
2. Analyse des Unternehmensmodells
   Die Analyse der erhobenen Informationen muß sehr sorgfältig vorgenommen werden. Durch die Einführung von CASE-Technologien soll nicht nur die Softwareentwicklung im Detail, sondern auch die Gesamtorganisation unterstützt werden. Dazu ist es notwendig, das praktizierte Vorgehen sowohl global als auch im Detail zu analysieren. Ein globaler Aspekt ist beispielsweise das der Praxis zu-

grundeliegende Vorgehensmodell, die zum Einsatz kommenden Methoden sind Detailaspekte.

3. Synthese des Unternehmensmodells
   Sind die Schwachstellen des derzeit praktizierten Unternehmensmodells analysiert, werden in einem nächsten Schritt die Verbesserungen definiert. Bezüglich der organisatorischen Aspekte werden Veränderungen an der Ablauf- und Aufbauorganisation des Unternehmens vorgeschlagen. Die technischen Aspekte werden durch die Auswahl von Software- und Hardwareplattformen berücksichtigt.
4. Praxiseinführung
   Das neue Unternehmensmodell kann nicht in einem Schritt eingeführt werden. Derzeitige Überlegungen gehen davon aus, die vorgeschlagenen Verbesserungen zunächst in einem Teilbereich des Unternehmens einzuführen. Dazu wird eine ausgewählte Gruppe von Mitarbeitern in den einzusetzenden Methoden und Werkzeugen geschult. Es folgt der Einsatz des erstellten Konzepts in einem Demonstrationsprojekt. Die Erfahrungen daraus werden genutzt, um das Konzept vor einer breiten Einführung noch einmal zu validieren und eventuell zu verbessern.

Dieses Vorgehen erfolgt - aufgrund des Querschnittscharakters des Arbeitsgebiets CASE-Einführung - in enger Kooperation mit anderen Arbeitsgebieten im Institut. Die Arbeiten zur Unternehmensmodellierung werden durch Konzepte und Werkzeuge des Bereichs PE gestützt. Bei der Synthese werden Resultate zum Software-Fabrik-Engineering aus dem Bereich ISIS verwendet.

Durchgeführte Arbeiten und Ergebnisse

Mit der Ausarbeitung der beschriebenen Vorgehensweise zur CASE-Einführung wurde begonnen.

Im einzelnen wurde an folgenden Teilaufgaben gearbeitet:

- **Konzepte und Werkzeuge zur Beschreibung des status quo**
  Dazu wurden mehrere Fragebogen-Sätze erstellt, Auswertungsprozeduren für Fragebögen entworfen und Werkzeuge für die Erfassung und Auswertung der Informationen entwickelt.
- **Konzepte für die Analyse des Unternehmensmodells**
  Diese Arbeiten umfassen die Entwicklung von Konzepten und Methoden, um die Anforderungen an die Organisation, die einzuführenden Methoden und Werkzeuge sowie an die Qualifikation der Mitarbeiter zu erheben.
- **Konzepte für die Synthese des Unternehmensmodells**
  Diese Arbeiten umfassen die Entwicklung von Konzepten und Methoden zur Auswahl von potentiellen Methoden und Werkzeugen und die Vorbereitung der dazu notwendigen Schulungen.

Ein erstes Anwendungsprojekt im Arbeitsgebiet CASE-Einführung wurde für das Software-Industrie Support Zentrum (SISZ) in Dortmund durchgeführt.

In Kooperation mit anderen Unternehmen wurde eine Methode zur Beratung von kleinen und mittelständischen Softwarehäusern bei der Einführung von Software-Entwicklungsumgebungen für objektorientierte Softwareentwicklung erstellt. Außerdem wurden Materialien für einen Seminarzyklus ausgearbeitet. Die Arbeiten sind Ende 1992 abgeschlossen worden.

2.2.7 CASE-Kompetenzzentrum

Aufgabenstellung und Ziele

Derzeit wird am Markt eine Vielzahl von CASE-Werkzeugen angeboten. Für den Softwareentwickler ist diese Fülle oft schwer überschaubar. Prospektmaterialien der CASE-Hersteller und auch Werkzeugübersichten verschiedener Autoren eignen sich nur bedingt zur Auswahl, weil die Beschreibungen häufig zu allgemein sind und die Bewertungskriterien auf den Einzelfall vielfach nicht zutreffen.

Im Rahmen eines CASE-Kompetenzzentrums soll daher potentiellen CASE-Anwendern ein Überblick über existierende Werkzeuge und Umgebungen geboten werden.

Vorgehensweise

Das CASE-Kompetenzzentrum soll eine Reihe von Softwareentwicklungswerkzeugen und -umgebungen auf verschiedenen Hardware- und Softwareplattformen demonstrieren können.

Das ISST kann aufgrund seiner neutralen Position herstellerunabhängige Bewertungen anbieten. Zu diesem Zweck wird eine Werkzeug-Datenbank aufgebaut. Diese soll - anders als die bisher bekannten Tool-Datenbanken, die lediglich Herstellerinformationen zusammenfassen - auf der Grundlage anforderungsspezifischer Bewertungen Informationen zur Verfügung zu stellen, die die Auswahl erleichtern. Damit Softwareentwickler, die nicht mit CASE-Werkzeugen vertraut sind, die Arbeit in einer solchen Umgebung kennenlernen, werden spezielle Schulungen angeboten.

Mit dem vom ISST und seinen Kooperationspartnern entwickelten Technologien und Systemen soll im Institut eine Software-Fabrik IRIN (ISST Research Instance) als „show case“ aufgebaut werden, die dieses Leistungsspektrum demonstriert.

Durchgeführte Arbeiten und Ergebnisse

Die Konzipierung des Kompetenzzentrums wurde abgeschlossen. Erste Werkzeuge wurden beschafft und mit deren Bewertung begonnen.

Im Rahmen des Anwendungsprojekts für das SISZ wurden Werkzeuge für die objektorientierte Softwareentwicklung bewertet.

Veröffentlichungen

[Dew92a]; [Dew92b]; [DEL92]

2.3 Integrierte Software-Infrastrukturen

2.3.1 Zielsetzung und konzeptionelle Grundlagen

Softwaresysteme zeichnen sich häufig durch einen hohen Komplexitätsgrad aus. Derzeit in der Entwicklung befindliche sowie zukünftige Systeme werden durch den Zwang zur Integration in ihrer Komplexität noch weiter zunehmen. Komplexe Systeme lassen sich durch die folgenden Eigenschaften charakterisieren:

- Sie sind in der Regel verteilte Systeme.
- Sie unterstützen verschiedene Kategorien von Benutzern mit zum Teil völlig verschiedenen Anforderungen.
- Sie sind langlebige Wirtschaftsgüter.
- Sie sind aus bereits existierenden und neu entwickelten Komponenten zusammengesetzt.

Um die Langlebigkeit von Softwaresystemen sicherzustellen, müssen bei deren Entwicklung, Weiterentwicklung und Pflege ingenieurtechnische Prinzipien angewendet werden. Dies ist insbesondere vor dem Hintergrund von Bedeutung, daß diese Softwaresysteme einen wesentlichen Anteil der gesamten informations- und kommunikationstechnischen Infrastruktur eines Unternehmens ausmachen.

Wesentliche Prinzipien, die bei der Entwicklung solcher Softwaresysteme zum Einsatz kommen, sind Strukturierung, Standardisierung und Evolutionsunterstützung. Die Identifizierung von regulären, d.h. strukturell einfachen, in der stets gleichen Form vorkommenden Infrastrukturen stellt einen wesentlichen Schritt in diese Richtung dar. Die Zusammenführung von Systemen und Komponenten zu komplexeren Strukturen und deren Integration in reguläre Software-Infrastrukturen bietet die beste

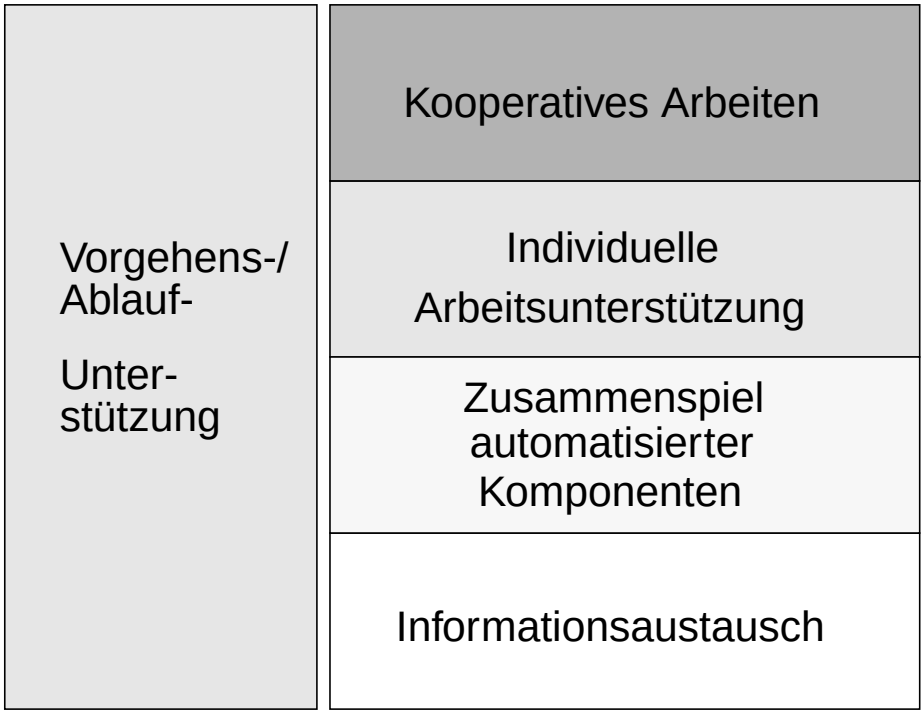


Abb. 3: Integrationsebenen in integrierten Infrastrukturen

Gewähr dafür, daß ein System über mehrere Jahrzehnte hinweg mit vertretbarem Aufwand an sich ändernde Randbedingungen angepaßt werden kann.

Der Bereich ISIS des ISST befaßt sich mit der Entwicklung von Konzepten, Architekturen, Techniken und Methoden sowie mit deren Umsetzung in konkrete Systeme und Umgebungen auf der Grundlage von Software-Infrastrukturen. Dabei ist das ISST an der Entwicklung bedeutsamer internationaler Lösungen beteiligt. Ein Beispiel dafür ist das Kernel/2r-System, eine Integrationsplattform, die im Rahmen des ESF-Projekts entwickelt wurde. Auf deren Basis lassen sich mit vergleichsweise geringem Aufwand äußerst flexible und evolutionsfähige Systeme entwickeln.

Software-Infrastrukturen werden als aufeinander abgestimmte Integrationsrahmen und -mechanismen verstanden, die

- kooperatives Arbeiten,
- individuelles, rechnergestütztes Arbeiten,
- das Zusammenspiel von Komponenten und
- den Informationsaustausch zwischen verschiedenen Systemen unterstützen.

Begleitend zu diesen vier Ebenen ist eine Vorgehens- oder Ablaufunterstützung zu sehen.

Diese Sichtweise hat sich im Verlauf des ESF-Projekts herausgebildet. Die dabei konzipierten und realisierten Software-Infrastrukturen sind nicht auf den Anwendungsbereich Softwareentwicklung beschränkt, sondern weitestgehend auf komplexe Softwaresysteme für sehr verschiedene Anwendungsbereiche übertragbar.

In komplexen Infrastrukturen der verschiedensten Anwendungsbereiche bestehen häufig Wechselbeziehungen zwischen den Arbeiten verschiedener Benutzer. Die Spannbreite der Abhängigkeiten reicht dabei von der Benachrichtigung eines anderen über ein Ereignis (z.B. über die Erledigung einer Aufgabe) bis hin zu verzahnten Interaktionen mehrerer Personen (z.B. gemeinsame Dokumentenbearbeitung - Joint Editing). Dafür sind grundlegende Mechanismen erforderlich, die sowohl Kommunikationsunterstützung bieten als auch die Arbeiten auf den verschiedensten Ebenen synchronisieren.

Die individuelle Arbeitsunterstützung beinhaltet u. a. eine Vereinheitlichung der interaktiven Schnittstelle von Werkzeugen hin zu einem einheitlichen „look and feel“. Dies ist aber nur ein Aspekt der Unterstützung des Mensch-Maschine-Dialogs. Die in den Arbeiten des Bereichs angestrebte Unterstützung umfaßt auch die Arbeitsorganisation des einzelnen Benutzers. Ein Beispiel dafür ist die Integration von Agenda-Konzepten[6]. Hier reicht die Palette von einfachen Benachrichtigungen, z.B. in Form von Einträgen in integrierte Terminkalender, bis hin zum Erzwingen der strikten Einhaltung von Arbeitsvorschriften.

Das Zusammenspiel von Komponenten und Systemen wird durch deren Interoperabilität sichergestellt. Die Integration existierender Komponenten in eine Umgebung führt in der Regel zu wechselseitigen Ein- und Ausgaben. Dabei tritt gewöhnlich das Problem auf, daß die Komponenten unterschiedliche Datenformate benutzen. Datentransformationen stellen somit einen wesentlichen Aspekt der Integrationsunterstützung auf dieser Ebene dar. Hinzu kommen Dinge wie die Transparenz der Verteilung von Komponenten oder Daten oder der Transaktionsschutz bei der Ausführung komponentenübergreifender Operationen.

Auf der untersten Ebene, dem Informationsaustausch zwischen den Komponenten innerhalb eines Systems, geht es um die Einbeziehung der Kommunikationsinfrastruktur. Die Basis dafür stellen entsprechende Referenzmodelle (z.B. ISO/OSI, BERKOM) und Kommunikationssubsysteme dar.

6. In diesem Kontext bedeutet Agenda eine Liste von Aufgaben, die von einer Person im Rahmen ihrer Tätigkeiten auszuführen ist.

2.3.2 Arbeitsgebiete

Der Bereichs ISIS ist in verschiedene Arbeitsgebiete unterteilt. Diese Gliederung folgt bewußt nicht der oben beschriebenen Aufteilung eines Systems in die Integrationsebenen, sondern reflektiert funktionale Gesichtspunkte.

- **Software-Fabrik-Engineering**
  Dieses Arbeitsgebiet umfaßt die Bereitstellung von Konzepten zur Erstellung, Einführung und zum Betreiben von Software-Fabriken sowie entsprechender Infrastrukturen. Das Leistungsspektrum reicht von der Ermittlung und Festlegung der Anforderungen an eine Software-Fabrik und der Präzisierung der durch sie zu unterstützenden Prozesse über die Integration von Werkzeugen zu einer Software-Fabrik bis hin zur Integration verschiedener Fabriken.
- **Daten- und Objektmanagement**
  Hierunter fallen Arbeiten zur Konzipierung und Entwicklung neuartiger, aus Sicht der Applikationen angemessener Datenmanagementsysteme. Es werden Konzepte zur Objektorientierung von relationalen Datenbanken sowie von ECMA-PCTE/OMS erarbeitet und umgesetzt. Ausgangspunkt hierfür sind Erfahrungen im Umgang mit konventionellen Datenbanksystemen und Nicht-Standard-Datenbanksystemen bzw. Objektverwaltungssystemen. Einen weiteren Schwerpunkt bilden Ansätze und Lösungen zur Koexistenz von Datenbanksystemen in komplexen Umgebungen.
- **Systemmanagement großer Netze**
  Zum Systemmanagement großer Netze gehören alle höheren administrativen Tätigkeiten in mittleren bis sehr großen Netzen von Computern. Diese Administration bezieht sich sowohl auf die Software als auch auf die Benutzergruppen. Die Beratung bei der Entwicklung und beim Einsatz von Systemmanagement-Konzepten ist ebenso Arbeitsinhalt wie die Entwicklung und Bereitstellung von Werkzeugen zur Managementunterstützung. Ein Beispiel für den letztgenannten Punkt sind Werkzeuge zur Verteilung und zur Installation von Software-Systemen in heterogenen Netzen.
- **Multimedia-/Hypermedia-Systeme**
  Dieses Arbeitsgebiet umfaßt die Bereitstellung von multimedialen Mitteln zur Unterstützung von verteilten Arbeitsgruppen sowie die Konzipierung und Realisierung von Hypermedia-Systemen. Schwerpunkte stellen multimediale Oberflächen zur Unterstützung der Arbeit verteilter Gruppen auf der Basis der Breitbandkommunikation dar.

Eine wesentliche Aufgabe des Bereichs ISIS ist der Technologietransfer, der auf der Basis von Forschungs- und Entwicklungsprojekten erfolgt. Diese Projekte lassen sich in der Regel leichter einem der hier genannten Schwerpunkte als einer Integrationsebene zuordnen, da sie sich gewöhnlich über mehrere Integrationsebenen erstrecken. Auch wenn sie ihren Problemschwerpunkt auf einer der Integrationsebenen haben.

2.3.3 Software-Fabrik-Engineering

Zielsetzung

Das Arbeitsgebiet Software-Fabrik-Engineering beschäftigt sich mit der Entwicklung von Infrastrukturen zur Unterstützung der industriellen, ingenieurmäßigen Produktion von Software. Ziel ist die systematische Entwicklung von Software auf hohem Qualitätsniveau bei hoher Produktivität.

Dabei liegt das Hauptaugenmerk nicht auf der Entwicklung und Bereitstellung von Werkzeugen zur Softwareentwicklung (s. dazu Abschnitt 2.2), sondern auf Infrastrukturen zur Unterstützung vor allem der organisatorischen Abläufe und der Integration in existierende Umgebungen. Es ist längst nachgewiesen, daß der koordinierte Transfer von Dokumenten und deren abgestimmte Bearbeitung in komplexen Organisationen ein massives Problem darstellen.

Die industrielle Softwareproduktion beinhaltet nicht nur die reine Entwicklung, sondern auch Produktionsplanung, Qualitätssicherung, Vertriebsunterstützung, Fehlerbehebung etc. Dies verdeutlicht, daß es sich auch hier um eine komplexe Organisation handelt, die entsprechender Unterstützung bedarf.

Als Software-Fabrik wird der Unternehmensbereich bezeichnet, der für die Bereitstellung und Pflege aller Softwaresysteme eines Unternehmens verantwortlich ist. Typische Aufgabenbereiche in der Software-Fabrik sind die Softwareentwicklung, die Auswahl und der Einsatz lizenzierter Softwarepakete, die Integration und Weiterentwicklung der eingesetzten Softwaresysteme und die Betreuung der Anwender.

Externe Beziehungen pflegt eine Software-Fabrik zu ihren Kunden, den Anwendern der bereitgestellten Softwaresysteme, den Lieferanten von Hardware- und Softwaresystemen sowie zu Dienstleistungsunternehmen, die spezielle Kenntnisse oder zusätzliche Kapazitäten bereitstellen.

Vorgehensweise

Die Arbeiten zum Software-Fabrik-Engineering erfolgen im wesentlichen im Rahmen des ESF-Projekts. Dessen Ziel ist die Entwicklung von Software-Engineering-Technologien zur industriellen Produktion qualitativ hochwertiger Software. Der Entwicklungsprozeß soll weitestgehend automatisiert und rationalisiert werden, die Standardisierung von Methoden und Schnittstellen vorangetrieben und Softwarekomponenten bzw. Teilsysteme sollen wiederverwendet werden. Im Mittelpunkt der Überlegungen steht dabei der Mensch, der an der Softwareentwicklung in unterschiedlichen Rollen beteiligt ist.

Im ESF-Konsortium sind deutsche, französische und britische Firmen sowie die Universität Dortmund vertreten. Das ISST ist Subkontraktor der Universität Dortmund.

Das Projekt ist mit einer Gesamtlaufzeit von 10 Jahren geplant und derzeit bis 1994 bewilligt.

ESF hat ein Referenzmodell entwickelt (ESF Conceptual Reference Model - CoRe), das die Kommunikationsbeziehungen in einer Software-Fabrik strukturiert. Dieses fungiert als Grundlage zur Beschreibung existierender Systeme und Prozesse im Sinne eines Netzes von organisatorischen, interpersonellen und automatisierten Arbeitsabläufen.

CoRe unterscheidet dabei zwischen InterWorking (Kooperation von Personen), InterAction (Mensch-Maschine-Dialog), InterOperation (Interoperation von Werkzeugen), InterFunction (Werkzeug-Plattform-Dialog) und InterConnection (Kommunikation zwischen verschiedenen Plattformen).

CoRe bildet die konzeptionelle Basis für die Entwicklung von Integrationstechnologien, wie z.B. der Integrationsplattformen Kernel/1 und Kernel/2r sowie eine Reihe von Produkten der Industriepartner in ESF.

Ergebnisse

Das ISST ist maßgeblich an der Entwicklung des ESF-Referenzmodells CoRe und der CoRe-konformen Instanz einer Integrationsplattform im Rahmen des Kernel/2r-Subprojekts beteiligt.

Der Kernel/2r wurde in Kooperation mit der Universität Dortmund und der Gesellschaft für Softwaretechnologie mbH (STZ), Dortmund, entwickelt.

Der Kernel/2r stellt ein komplettes Framework dar und besteht aus:

- CorMan (Coordination Manager), der Prozeßmanagement- bzw. InterWorking-Komponente (ca. 250.000 Code-Zeilen)

  CorMan unterstützt die Definition von Softwareprozessen und die Teamarbeit. CorMan besteht aus einer Menge von Werkzeugen zur Prozeßmodellierung, zur Analyse, Simulation und Darstellung von Modellen und verwendet die Prozeßmodellierungssprache FUNSOFT (s.a. Abschnitt 2.4).

- WHOW (We Help Organize Work), der InterAction-Komponente (ca. 30.000 Code-Zeilen)

  Diese Komponente unterstützt die Arbeitsorganisation und die verschiedenen Arbeitsprozesse eines Benutzers. Jeder Anwender verfügt über eine eigene WHOW-Instanz. WHOWs überwachen diejenigen Teilnetze des gesamten Prozeßmodells, die nur Aktivitäten und Aktionen für jeweils einen Benutzer beinhalten. WHOWs sind (in der ESF-Terminologie) Anwender-Interaktionsumgebungen, welche einem Anwender eine graphische Oberfläche zur Unterstützung seiner Arbeiten bereitstellen. Eine WHOW beinhaltet unter anderem eine Multi-View-Agenda (Projektsicht, Terminsicht, usw.) zur Durchführung von Aktivitäten

und Aktionen, Aktivität-Objekt-Werkzeug-Bindungen und Benachrichtigungen sowie zur Anpassung an individuelle Benutzerwünsche.

- MUSE (Multi component integration Services), dem Software Bus bzw. der Inter-Operation-Komponente (ca. 130.000 Code-Zeilen)

  MUSE ermöglicht die Zusammenarbeit der verschiedenen Werkzeuge und fungiert als Kommunikationskanal zwischen CorMan und den verschiedenen WHOWs. MUSE unterstützt die Kooperation über Interfaces, die das Einfügen neuer Service-Komponenten gestatten. Ferner unterstützt MUSE die Verkettung automatisierter Aktionsfolgen, welche auf die verschiedenen, an unterschiedlichen Orten residierenden Service-Komponenten zugreifen können.

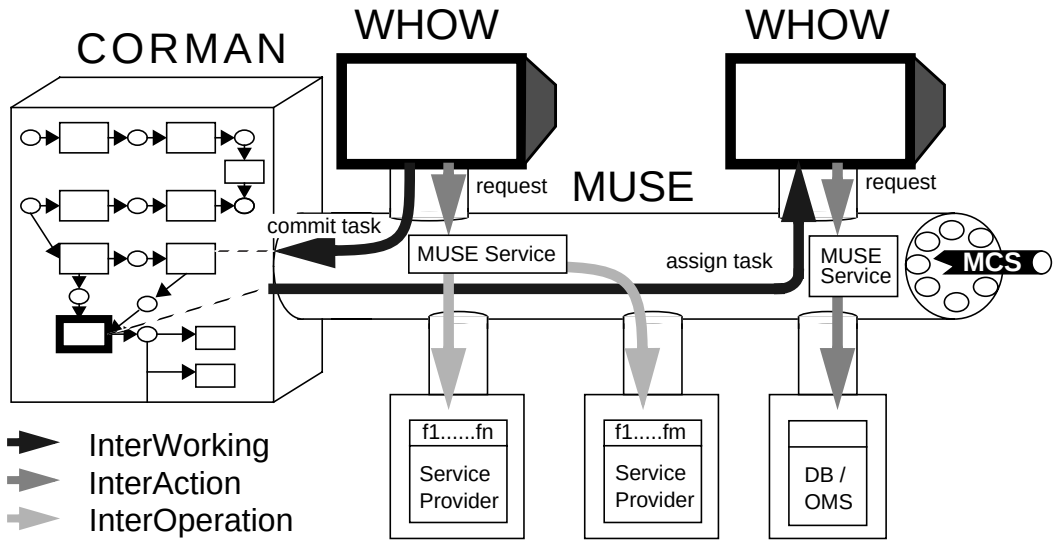


Abb. 4: Integrations-plattform Kernel/2r

- MCS (Multi Communication System), der InterConnection-Komponente (Verbindungsschicht, ca. 20.000 Code-Zeilen)

  MCS bietet Verteilungs-Transparenz und Unterstützung heterogener Rechnerumgebungen über lokale und globale Netzwerke unter Berücksichtigung der ISO-Standards. Die Kommunikation zwischen den verschiedenen Rechnersystemen basiert auf den X.500 Name and Location Services und auf einem Message Handling Systemmodell, welches eine einheitliche Schnittstelle zu den Kommunikationsdiensten TCP/IP, X.400 bzw. IPC bereitstellt.

Der Kernel/2r ist in C geschrieben und läuft auf Sun Microsystems Workstations. Das Framework wurde innerhalb von 18 Monaten implementiert, basierend auf schon existierenden Komponenten. Der Kernel/2r besitzt keine spezielle Datenhaltungs-Komponente, stattdessen können beliebige Datenbanken (wie z.B. das relationale DBMS Oracle) als Service-Komponenten (Populatoren) in das Framework integriert werden.

Derzeit wird an der Realisierung einer Kernel/2r-basierten Software-Fabrik IRIN für das ISST gearbeitet (IRIN = ISST Research Instance).

Veröffentlichungen

[ADH+]; [ADHW92]; [HAD93]; [Hol92a]; [Hol92c]; [Hol92f]; [Hol92g]; [Hol93]; [Web92b]; [Web92i]; [Web92f]

2.3.4 Daten- und Objektmanagement

Zielsetzung

In modernen Unternehmen sind heute häufig drei bis vier verschiedene Datenbanksysteme neben einer Reihe von Dateisystemen der eingesetzten Rechner anzutreffen. Dabei reicht die Bandbreite der benutzten Speichertechnologien von konventionellen Disketten und Festplatten bis hin zu CD-ROM und optischen Wechselplatten. Genauso unterschiedlich wie die eingesetzten Technologien sind auch die darauf basierenden Anwendungen.

Damit ein Anwender seine Tätigkeit ausführen kann, muß er in der Regel wissen, in welchem Datenbanksystem sich die von ihm benötigten Informationen befinden, welcher Art die Informationen sind (z.B. Tabellen, Text, Grafik, Bilder usw.), wie sie strukturiert sind u. dgl.

Mit den Arbeiten auf dem Gebiet Datenhaltung soll der Benutzer davon entlastet werden, eine Fülle von - auch irrelevanten - Details beachten zu müssen. Ziel ist die Transparenz der eingesetzten Datenbanktechnologie.

Vorgehensweise

Objektorientierte Datenbanken stellen einen wesentlichen Schritt in diese Richtung dar. Sie unterstützen die Verwaltung der verschiedensten - insbesondere auch komplexer - Datentypen in einem System und bieten gleichzeitig vereinheitlichte Zugriffsoperationen auf die unterschiedlichen Daten an.

Die objektorientierte Datenbanktechnologie stellt deshalb einen Schwerpunkt dieses Arbeitsgebietes dar.

Die angestrebte Transparenz wird mit Objektorientierung allein aber nicht erreicht, da objektorientierte Datenbanken letztlich auch nur ein Glied in einer Reihe von Datenbanktechnologien sind, die von Unternehmen eingesetzt werden, und nicht die Integration anderer, bereits existierender Datenbanktechnologien beinhalten.

Inzwischen ist in den Unternehmen der Trend zur Integration der bereits existierenden Informationsinseln in ein unternehmensweites Informationsnetz zu beobachten. Bedingt durch die unterschiedliche Entwicklung verschiedener Unternehmensbereiche resultiert daraus die Forderung nach der Integration unterschiedlicher Datenbanktechnologien zu einem Gesamtsystem, das dem Benutzer in einheitlicher Form Datenhaltungsdienste für die verschiedensten Datentypen bereitstellt und dabei Aspekte wie Verteiltheit etc. vor dem Benutzer verbirgt. Für einen entsprechenden Lösungsansatz haben wir den Begriff „Object Management Machine“ geprägt.

Eine wesentliche Basis für die Entwicklung von Object Management Machines ist die Realisierung von Koexistenz-Konzepten für verschiedene Datenbanktechnologien. Der hier verfolgte Ansatz geht davon aus, daß logische Beziehungen, wie beispielsweise Konsistenzbedingungen, zwischen Daten bestehen, die in verschiedenen Datenbanken abgelegt sind. Als Integrationsplattform steht der im vorangegangenen Abschnitt beschriebene Kernel/2r zur Verfügung.

Zu dem Aspekt der Vereinheitlichung von Zugriffsoperationen auf komplexe Datentypen werden die an der Universität Dortmund begonnenen Arbeiten an dem objektorientierten Datenbanksystem InORM weitergeführt. Hier liegt der Schwerpunkt auf der Weiterentwicklung des Objektmodells und der zugehörigen Abfragesprache, die eine Erweiterung des relationalen Ansatzes darstellt.

Die dabei gewonnenen Erkenntnisse und Erfahrungen fließen in das ECMA-PCTE-Projekt ein. PCTE steht für eine Datenbank-zentrierte Entwicklungsumgebung. Den Kern bildet das PCTE/OMS (Object Management System), dessen Schnittstelle standardisiert ist. Neben der Implementierung des PCTE-Standards durch die Industriepartner (bisher existieren nur Vorläufer-Versionen, die dem Standard nicht voll entsprechen) stellen die Arbeiten des ISST zur vollen Objektorientierung von PCTE, d.h. Einbeziehung von Methoden und Schaffung von Definitionsmöglichkeiten für komplexe Objekte, sowie zur Entwicklung von Konzepten für die Koexistenz von PCTE/OMS mit anderen Datenhaltungssystemen weitere Schwerpunkte des Projekts dar.

Ergebnisse

Um den Anforderungen der ständig wachsenden Zahl und der neuen Arten von Datenbankanwendungen, wie Softwareentwicklung oder Multimedia-Systemen, gerecht zu werden, ist das objektorientierte Datenbanksystem InORM weiterentwikkelt worden.

Das dabei erarbeitete Datenmodell von InORM ist ein erweitertes relationales Datenmodell und enthält einige entscheidende Konzepte, die man in ähnlicher Weise auch bei anderen objektorientierten Systemen findet. In diesem Datenmodell gibt es ein Typsystem, in dem ein Typ - vergleichbar einer Klasse anderer objektorientierter Systeme - und dessen Instanzen getrennt deklariert werden. Es existiert eine Typhierarchie, die über ein Vererbungskonzept gebildet wird. Während innerhalb des

Typ- und Objektsystems die gemeinsame Verwendung von Objekten unterstützt wird, können Daten, die nicht geteilt werden, auch allein über ihre Werte repräsentiert werden. Die Mischung aus Objekt- und Wertsemantik für Daten verleiht dem Datenmodell seine Mächtigkeit. Das Typsystem expandiert nicht so schnell und das gesamte Datenbanksystem erreicht eine höhere Leistung als bei einer reinen Objektsemantik. Die benötigten Operationen und Befehle werden in Form einer SQL-ähnlichen Sprache (EQL) zur Verfügung gestellt. EQL ist eine objektorientierte Erweiterung von SQL. Sie erweitert die Abfragesprache um die Definition und Manipulation von Typen, Instanzen und Komplex-Werten, zusammen mit erweiterbaren Funktionen und Operationen, welche den Typen zugeordnet sind. Als Typen sind sowohl Grundtypen als auch abstrakte Datentypen möglich. Die Abfragesprache stellt die Operationen auf Objekten und relationale Operationen in einer integrierten Umgebung zur Verfügung.

Im Rahmen des ECMA-PCTE-Projekts ist ein Konzept für die Erweiterung des PCTE/OMS zu einem voll objektorientierten System OOPCTE entwickelt worden; die Ergebnisse der Arbeit sind in einer Studie zusammengefaßt.

Der ausgearbeitete Vorschlag für die Behandlung von Methoden in OOPCTE lehnt sich eng an das Schema-Definition-Set (SDS) Konzept des PCTE/OMS an. Er versucht, dessen Semantik auch auf die Ausführung von Methoden zu übertragen. Die Methoden und ihre Ausführung werden in OOPCTE integriert, um ein homogenes System zu erhalten. Es wird dabei keine endgültige Festlegung bezüglich der Implementierung vorgenommen, vielmehr werden einige sinnvolle Möglichkeiten beschrieben.

OOPCTE ist gegenüber PCTE um Möglichkeiten zur Definition komplexer Objekte (Tupel, Listen, Mengen, ...) erweitert worden.

Das Konzept der Object Management Machines ist in einer ersten Version niedergelegt worden [HW92]. Die Arbeiten zu Koexistenz-Konzepten sind nach dem Abschluß der OOPCTE-Studie intensiviert worden. Sie bilden einen Arbeitsschwerpunkt für die folgenden Jahre.

Veröffentlichungen

[HEV92]; [HW92]; [XF93]

2.3.5 Multimedia-/Hypermedia-Systeme und kooperatives Arbeiten

Zielsetzung

Die rasche Entwicklung auf den Gebieten Hardware- und Kommunikationstechnologie hat längst die Voraussetzungen für den Austausch multimedialer Informationen zwischen verteilt arbeitenden Gruppen geschaffen. Anwendungen wie z.B. die Durchführung von Video-Konferenzen auf PC-Basis sind bereits am Markt verfügbar; auch auf Messen und bei den Anbietern ist ein deutlicher Trend zu Multimedia-Anwendungen erkennbar. Dabei steht zu befürchten, daß dieser Multimedia-Euphorie eine ähnliche Ernüchterung folgt, wie sie vor einiger Zeit bei den CASE-Werkzeugen beobachtet werden konnte. Multimedia-Systeme sind zwar von ihrer Oberfläche her ansprechend und ermöglichen die Erstellung sehr komplexer Anwendungen; der damit verbundene Aufwand (bezüglich der Erstellungszeiten und der benötigten Ressourcen) wird jedoch häufig genauso unterschätzt wie die zur Benutzung erforderlichen Fähigkeiten. Bei Anwendungen wie Video-Konferenzen stellen zudem die Betriebskosten einen beachtlichen Faktor dar.

Das Ziel der Arbeiten auf dem Gebiet „Multimedia-/Hypermedia-Systeme und kooperatives Arbeiten“ besteht in der Entwicklung von Konzepten für entsprechende Software-Infrastrukturen und deren Realisierung. Dabei wird, um das weite Anwendungsfeld einzuschränken, die Unterstützung kooperativen Arbeitens betont.

Vorgehensweise

Das ISST bietet wegen seiner Struktur für die Untersuchung kooperativer Arbeitsprozesse günstige Voraussetzungen: Da sich das Institut, insbesondere der Bereich ISIS, über die Standorte Berlin und Dortmund verteilt, sind Kooperationen über große Entfernungen an der Tagesordnung. Auch in Zukunft werden Projekte von den Arbeitsgruppen in Berlin und Dortmund gemeinsam bearbeitet.

Am Anfang steht also die Analyse der Bedürfnisse im eigenen Umfeld. Davon ausgehend werden Infrastrukturen konzipiert und realisiert, die entweder auf existierenden Standards basieren oder sich an zukünftigen orientieren. Kernel/2r stellt auch hier einen Integrationsrahmen für die Entwicklungen zur Verfügung.

Da hinter der Vorgangsunterstützung beim Kernel/2r (s. Abschnitt 2.3.3) die Unterstützung *asynchroner* kooperativer Vorgänge steht, kann dazu auf existierende Lösungen zurückgegriffen werden. Deshalb liegt der Schwerpunkt bei den Arbeiten hier auf der *synchronen* Bearbeitung arbeitsteiliger Vorgänge, wie z.B. Konferenzen oder Beratungen.

Ergebnisse

Es wurde geprüft, inwiefern eine technisch unterstützte Multimedia-Kommunikation für eine effektivere Zusammenarbeit zwischen Berlin und Dortmund sinnvoll sein könnte. Dazu wurden einige Szenarien aus verteilt durchgeführten Projekten, z.B. dem ECMA-PCTE-Projekt, analysiert und als Grundlage für die weiteren Betrachtungen verwendet.

Dabei wurde festgestellt, daß Videokonferenzen für bestimmte Typen von Gesprächen nicht geeignet sind: Wo es um Motivationen, Haltungen oder Kritik geht, sind elektronische Kommunikationsmittel aufgrund der eingeschränkten Übertragungsmöglichkeiten - so läßt sich z.B. Atmosphäre nicht übermitteln - unzureichend. Liegt der Zweck der Kontakte in technischen Abstimmungen oder Terminabsprachen, so erweisen sich Videokonferenzen dagegen als ein adäquates Mittel für die Zusammenarbeit zwischen räumlich getrennten Gruppen.

Weiterhin stellte sich heraus, daß eine reine Videokonferenz vielfach wenig hilfreich ist; erst wenn weitere Werkzeuge hinzukommen, gewinnt sie an praktischem Nutzen. Als hilfreiche Ergänzung hat sich die Integration von Dokumenten - vom einfachen elektronischen Hochhalten bis zum gemeinsamen Modifizieren - sowie von Zeichentafeln und Diskussionsmanagern erwiesen.

Vor dem finanziellen Hintergrund (derzeit kostet eine Videokonferenz 900 DM pro Stunde über 2 Mbit-ISDN) ist der Aspekt Diskussionsmanagement besonders interessant. Durch eine koordinierte und zielgerichtete Diskussion läßt sich das Instrument Videokonferenz effektiver einsetzen.

Die Untersuchungsergebnisse sind in einen BERKOM-Projektantrag zur Entwicklung eines Videokonferenzsystems auf der Basis eines definierten Vorgehensmodells eingeflossen. Mit Vorarbeiten zum Projekt wurde begonnen.

Zur Unterstützung von Konferenzen, die primär auf elektronischer Post basieren, wurde ein Konzept zur elektronischen Postkoordinierung entworfen. Dieses kann vor allem dort angewandt werden, wo an einer zentralen Stelle globale Dienste für viele vernetzte Arbeitsplatzrechner angeboten werden. Beispiel dafür sind Rechnerbetriebe, die für große lokale Netze zuständig sind.

Veröffentlichungen

[Mes92a]; [Mes92b]

2.3.6 Systemmanagement großer Netze

Zielsetzung

Obwohl in den letzten Jahren die Anzahl verteilter Systeme in den verschiedensten Anwendungsbereichen drastisch zugenommen hat, ist die Situation in bezug auf die Systemverwaltung äußerst schlecht. Der Preisverfall bei der Rechentechnik, insbesondere auch bei der Vernetzung, führte dazu, daß zwar viele Arbeitsplätze bereits vernetzt sind; die entsprechende Unterstützung zur Systemverwaltung fehlt aber häufig völlig oder ist nur rudimentär vorhanden. Erst in letzter Zeit wird diese Problematik stärker beachtet.

Drei Einflußgrößen spielen bei der Verwaltung von großen Netzen für das Management dieser Systeme eine Rolle:

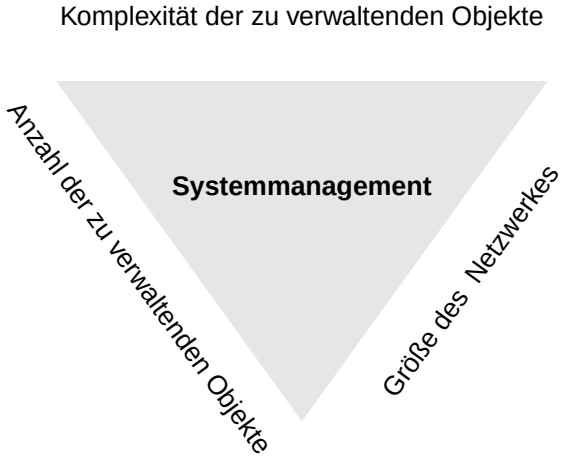


Abb. 5: Einflußgrößen beim Systemmanagement

Die zu verwaltenden Objekte (Managed Objects) beinhalten sowohl Systemdienste als auch Anwendungen. Mit steigenden Werten dieser Einflußparameter steigt der Aufwand für das Management solcher Systeme rapide an.

Während Personal Computer meist noch vom Anwender selbst verwaltet werden können, ist dies für vernetzte Systeme in der Regel nicht mehr der Fall. Man benötigt Systemmanager, die ein Netzwerk mit allen seinen Ressourcen verwalten. Um den mit den heutigen Möglichkeiten in der Regel unakzeptabel hohen Verwaltungsaufwand zu senken, müssen bessere Methoden und Werkzeuge entwickelt werden. Erste Ansätze, die Probleme zu lösen, werden heute im Netzwerk-Management realisiert. Dazu werden einheitliche Schnittstellen zum Verwalten von Managed Objects sowie komfortable Werkzeuge entwickelt. Die Werkzeuge beschränken sich allerdings auf Teilaufgaben oder einzelne Anwendungen.

Das Ziel des Arbeitsgebiets Systemmanagement besteht darin, ein durchgängiges Konzept zu entwickeln, das sowohl das Konfigurationsmanagement als auch das Installationsmanagement abdeckt. Damit werden Voraussetzungen für eine weitestgehende Automatisierung der Systemverwaltung geschaffen. Die Konzepte sollen als Software-Infrastrukturen für große verteilte Systeme realisiert werden. Dabei muß gewährleistet sein, daß sich die Lösungen nach Veränderungen im Unternehmen oder in einzelnen Bereichen einfach und kostengünstig anpassen lassen.

Ein effektives Management des Netzes und der darauf laufenden Anwendungssysteme ist z.B. auch für Software-Fabriken unverzichtbar, da für diese die Verteiltheit und eine große Zahl zusammenwirkender Prozesse und kooperierender Personen (Größenordnungen von mehreren Hundert bis zu mehreren Tausend sind in der Praxis üblich) charakteristisch sind.

Vorgehensweise

Als genereller Trend zeichnet sich beim Netzwerkmanagement die Ausrichtung auf OSI-konforme Standards wie CMIP (Common Management Interface Protocol) bzw. MIS/SMI (Management Information System/Systems Management Interface) ab. Zugehörige Produkte, soweit verfügbar, konzentrieren sich jedoch auf das Netzwerkmanagement und lassen das Systemmanagement - im Sinne einer integrierten, globalen Systemverwaltung - weitgehend offen.

An dieser Stelle setzen die Arbeiten des Arbeitsgebiets Systemmanagement an. Gängige Standards bilden ebenso wie beim Netzwerkmanagement die Basis für die Entwicklung und Umsetzung der Konzepte für das Systemmanagement. Dies gilt sowohl für die Management-Protokolle als auch für die zu unterstützenden Plattformen. Zu unterstützende Plattformen sind deshalb UNIX-Rechner, PCs und X-Terminals. Die zu realisierenden Dienste werden von UNIX-Rechnern zur Verfügung gestellt, da diese über die notwendigen Voraussetzungen für die Kommunikation und für das Prozeßmanagement verfügen.

Eine sinnvolle Forderung an allgemeingültige Lösungen für das Systemmanagement ist, daß diese von einem Modell des betrachteten Systems ausgehen. Daher bildet die Modellierung von großen, integrierten Systemumgebungen einen Schwerpunkt der Arbeiten. Weitere Schwerpunkte sind das Installationsmanagement und die Konfigurierung von Software.

Als Lösung wird eine zentrale, weitgehend automatisierte Systemverwaltung, die dezentral unterstützt wird, konzipiert. Die Tragfähigkeit der Konzepte wird durch die Entwicklung entsprechender Infrastrukturen und deren Einsatz in Pilotprojekten nachgewiesen.

Ergebnisse

Im Arbeitsgebiet Systemmanagement sind das Eigenforschungsprojekt SINUS sowie zwei Anwendungsprojekte bearbeitet worden.

Das Projekt SINUS umfaßt die folgenden Teilaufgaben:

- Grafische Modellierung von großen, integrierten Systemumgebungen,
- Erzeugung von Beschreibungen für den Installationsprozeß,
- Automatische Installation von Software,
- Überprüfung von Installationen,
- Einheitliche, integrierte Konfigurierung von Software.

Als erste Ergebnisse liegen Konzepte für das Installations- und Konfigurationsmanagement und darauf aufbauend Lösungen für das Konfigurationsmanagement von Rechnern und Benutzern vor.

In dem ersten Anwendungsprojekt wurde für das Landesamt für Informationstechnik (LIT) Berlin ein Konzept für Systemadministrationszentrum entwickelt. Dieses soll das im Aufbau befindliche Netz der Berliner Verwaltung mit mehreren tausend Arbeitsplätzen betreuen. In einer Studie wurden die Anforderungen an ein solches Systemmanagement definiert und ein Lösungsvorschlag erstellt. Die SINUS zugrundeliegenden Konzepte sind dabei mit eingeflossen.

Im zweiten Anwendungsprojekt wurde damit begonnen, für die Kreisverwaltung eines neuen Brandenburger Großkreises ein Konzept zu erarbeiten, das auf der Basis einer kommunikationsgestützten Systemlösung eine effiziente Verwaltung ermöglicht. Der Entwurf soll Modellcharakter auch für andere Kreisverwaltungen haben.

Veröffentlichungen

[Got92c]; [KMB92]

2.4 Prozeß-Engineering

Der Bereich Prozeß-Engineering befaßt sich mit der Entwicklung von Konzepten und Werkzeugen zur systematischen Modellierung, Analyse und Durchführungsunterstützung von Vorgängen unterschiedlicher Art sowie mit deren Anwendung auf praktisch relevante Probleme.

2.4.1 Systematisches Management von Vorgängen

Ziele und Aufgaben

Vorgänge finden in den verschiedensten Bereichen statt. Sie werden je nach ihrer Art auch als Geschäftsvorfälle, als Kommunikations- und Koordinationsprozesse oder als Entwicklungsprozesse bezeichnet. Im folgenden wird der Begriff *Vorgang* als Sammelbegriff verwendet.

Vorgänge bestehen aus einer Menge von Aktivitäten, die teilweise automatisiert ablaufen, teilweise die Mitwirkung von Personen erfordern. Innerhalb eines Vorgangs werden Aktivitäten in verschiedener Art und Weise angeordnet (manche Aktivitäten müssen in einer bestimmten Reihenfolge ausgeführt werden, andere können parallel nebeneinander stattfinden usw.). Die Beschreibung einzelner Aktivitäten (was ist zu tun, welche Werkzeuge werden eingesetzt), die Anordnung der Aktivitäten, die Definition von Verantwortlichkeiten für Aktivitäten sowie die Beschreibung des Aufbaus von Aktivitätsergebnissen werden zusammen als *Vorgangsmodell* bezeichnet.

Die systematische Behandlung von Vorgängen ist insbesondere im geschäftlichen Leben und bei der Entwicklung komplexer Produkte - dazu zählt auch Software - von erheblicher Bedeutung.

In der Praxis ist zu beobachten, daß viele Vorgänge zwar initial geplant werden, die ursprüngliche Planung jedoch schnell obsolet wird. Vorgänge laufen anders ab als geplant, die beteiligten Personen halten sich nicht an die Absprachen, Vorgänge überschreiten Zeit- und Kostenlimits, die Interaktionen zwischen den an einem Vorgang beteiligten Personen sind oft durch Kommunikationsschwierigkeiten geprägt.

Die systematische Behandlung von Vorgängen hat zum Ziel, die Vorgänge explizit zu machen, d.h. sie zunächst zu definieren. Vorgangsmodelle, als Definitionen von Vorgängen, tragen dazu bei, ein gemeinsames Verständnis von Vorgängen bei den beteiligten Personen zu wecken und Probleme diskutabel zu machen. Ist ein Vorgang beschrieben, kann das entsprechende Vorgangsmodell in vielfältiger Hinsicht überprüft werden. Solche Überprüfungen können Plausibilitäts- und Konsistenzbetrachtungen umfassen und auf spezielle Vorgangseigenschaften (Verklemmungen, kritische Pfade, Personalbedarf) abzielen.

Ein systematisches Vorgangsmanagement will sicherstellen, daß sich Vorgänge nur entsprechend den ihnen zugrundeliegenden Modellen entwickeln. Andererseits sollen Vorgangsmodelle jederzeit im nötigen Maße geändert und erweitert werden können. Die rechnergestützte Ausführung garantiert, daß automatisierte Aktivitäten ohne Verzögerungen begonnen werden und die am Vorgang beteiligten Personen jederzeit wissen, welche Aktivitäten sinnvollerweise auszuführen und welche zu bestimmten Zeitpunkten sinnlos sind.

Vorgangsmodellierung trägt auf diese Weise nicht nur dazu bei, Prozesse verständlich und planbar zu machen. Sie stellt zusätzlich eine wichtige Qualitätssicherungsmaßnahme dar. Qualität bezieht sich dabei nicht nur auf die Produkte (d.h. die Ergebnisse von Vorgängen), sondern auch auf die Vorgänge selbst. Konzentriert sich Qualitätssicherung ausschließlich auf die Ergebnisse eines Vorgangs, so werden unter Umständen Ressourcen verschwendet, weil Fehler zu spät erkannt werden. Wird dagegen bereits der Vorgang einer Qualitätsprüfung unterzogen, bestehen gute Chancen, daß auch das Endprodukt dieses Vorgangs qualitativ hochwertig ist.

Vorgehensweise

Grundlage für ein systematisches Vorgangsmanagement ist die Modellierung der betrachteten Vorgänge. Dazu werden Aufgaben, Dokumente, Rollen und Werkzeuge, die die grundlegenden Entitäten von Vorgangsmodellen darstellen, erhoben. Anschließend werden Vorgangsmodelle entwickelt, die die Ablauf und Aufbauorganisation beschreiben. Zur Definition dieser Vorgangsmodelle wird der FUNSOFT-Ansatz verwendet.

FUNSOFT-Netze sind eine Vorgangsmodellierungssprache, die eine einfache, graphische Modellierung von Vorgängen erlaubt. FUNSOFT-Netze gestatten (1) die explizite Darstellung und (2) die Analyse von Vorgangsmodellen. Darüber hinaus ermöglichen sie (3) eine rechnerbasierte Durchführung und Animation von Vorgängen. Dadurch wird eine optimale Unterstützung aller beteiligten Personen erreicht.

Der Ansatz zum Vorgangsmanagement auf der Basis von FUNSOFT-Netzen wird in verschiedenen Richtungen weiterentwickelt.

Zum einen werden Fragebögen- und Erhebungstechniken verbessert, mit denen von den jeweiligen Kompetenzträgern wichtige Informationen gewonnen werden können.

Darüber hinaus wird eine Bibliothek standardisierter Modellbausteine aufgebaut (wie z.B. das V-Modell als standardisiertes Vorgehensmodell für die Abwicklung von Datenverarbeitungsprojekten in öffentlichen Behörden; Methoden wie EUROMETHOD etc.). Diese Bausteine, für die bestimmte Qualitätseigenschaften nachgewiesen sind, werden als „Schablonen“ bei der Modellierung kundenspezifischer Vorgänge herangezogen und angepaßt. Dies reduziert den Erstellungsaufwand und gewährleistet, daß die zu definierenden Vorgänge bewährten Richtlinien folgen.

Ergebnisse

Die Werkzeuge zum Vorgangsmanagement auf der Basis von FUNSOFT-Netzen wurden weiter ausgebaut. Diese Werkzeuge, die alle genannten Aufgaben eines Managements von Vorgängen (Modellierung, Analyse und Ausführung) unterstützen, sind in der Vorgangsmanagement-Umgebung CorMan zusammengefaßt.

Der skizzierte Ansatz ist in Kundenprojekten praktisch angewandt worden (s. Abschnitt 2.4.2).

2.4.2 Management kundenspezifischer Geschäftsprozesse

Im Berichtszeitraum wurde der FUNSOFT-Netz-Ansatz innerhalb eines Projektes des Bundes- und der Länderfinanzministerien sowie im Rahmen eines Projektes mit der Kraftwerks-Simulator-Gesellschaft (KSG) zur Vorgangsmodellierung eingesetzt.

Darüber hinaus ist die Vorgangsmodellierung ein wesentlicher Bestandteil des ESF-Projekts (s. Abschnitt 2.3.4).

Modellierung für das Projekt FISCUS

Das Projekt FISCUS ist ein Gemeinschaftsprojekt des Bundes- und der Länderfinanzministerien zur Neukonzeption der EDV-technischen Abwicklung des Steuerwesens. Zur Zeit existieren in den Landesrechenzentren der verschiedenen Bundesländer Programmpakete zur Steuerabwicklung, die die einzelnen Länder eigenständig entwickelt haben. Da diese Systeme veraltet sind, sollen sie ersetzt werden. Das geplante neue System soll dabei gemeinschaftlich von den verschiedenen Ländern entwickelt werden.

Das ISST wirkt bei der Durchführung des FISCUS-Projektes mit. Im Rahmen einer Vorgangsmodellierung werden dabei Organisationsstrukturen definiert, die die Zusammenarbeit der verschiedenen Gremien (Koordinierungsausschuß, Assistenzgruppe, Referentengruppe, ausführende Gremien, systemtechnische Zentrale, Migrationsgruppe etc.) innerhalb des FISCUS-Projektes festlegen.

Neben den Kommunikationsbeziehungen zwischen verschiedenen Gremien werden auch einzelne Vorgehensweisen detaillierter modelliert. Vorgaben in Form expliziter Richtlinien garantieren die Qualitätssicherung der Arbeiten innerhalb des FISCUS-Projektes.

Modellierung für die Kraftwerks-Simulator-Gesellschaft (KSG)

Die KSG betreibt Simulatoren für verschiedene, im Bereich der Bundesrepublik Deutschland betriebene Kernkraftwerkstypen. Mängel an den Simulatoren erfordern sofortige Behebung. Die Anforderungen zu Simulatoränderungen werden in Form von Nachrüstungs- und Änderungsaufträgen (Discrepancy Reports) gestellt.

Da die einzelnen Simulatoren unterschiedliche Technologien realisieren, weichen auch die Vorgangsmodelle zur Bearbeitung der Aufträge teilweise erheblich voneinander ab. Zudem sind die Modelle nicht explizit beschrieben. Die feste Zuordnung von Mitarbeitern zu Simulatoren soll die erfolgreiche Bearbeitung von Discrepancy Reports sicherstellen.

Das ISST wurde damit beauftragt, ein Vorgangsmodell zur Bearbeitung von Discrepancy Reports verschiedener Simulatortypen zu entwerfen. Dies stellt eine wichtige Qualitätssicherungsmaßnahme dar. Neben der Dokumentation der Bearbeitung von Discrepancy Reports ist auch eine leichtere Einarbeitung von Mitarbeitern möglich. Außerdem können mit Modellanalysen Schwachstellen in den Vorgängen entdeckt und bei Bedarf eliminiert werden. Darüber hinaus werden Untersuchungen über den zeitlichen Ablauf der Vorgänge zur softwaretechnischen Simulatoranpassung/-erweiterung durchgeführt. Die Vorgänge werden so auf ihre Effizienz überprüft; Möglichkeiten zur Effizienzsteigerung können vorgeschlagen werden.

Veröffentlichungen

[Dei92a]; [Dei92b]; [Dei92c]; [DG92]; [DGW93]

2.5 Aus- und Weiterbildung

Der Bereich Qualifizierung führt verschiedene Maßnahmen im Bereich Weiterbildung und Technologietransfer durch. Das ISST bietet hier Leistungen an, die auf die Weiterqualifikation von Fachleuten und erfahrenen Anwendern abzielen und sich von traditionellen Schulungsangeboten abheben.

Die Experten der Fraunhofer-Einrichtung für Software- und Systemtechnik stellen ihre langjährigen Erfahrungen im Rahmen des deutschen Kursangebots zum „European Professional Software Engineering Program“ (EPSE) zur Verfügung.

Zum Technologietransfer wurde ein Projekt vorbereitet, das die Erstellung einer Software-Entwicklungsumgebung auf der Basis von Software-Prozeßtechnologie bei einem privaten Unternehmen zum Ziel hat. Dieses Projekt hat im Januar 1993 begonnen.

Das European Professional Software Engineering Program

Das EPSE-Programm ist ein Fortbildungsangebot an Softwareentwickler auf dem Gebiet des Software Engineering und der werkzeuggestützten Softwareentwicklung. Ausbildungsziel ist die Qualifikation zum „European Professional Software Engineer“. Ein europäisches Konsortium großer Softwarefirmen und führender Forschungsinstitutionen auf dem Gebiet der Softwaretechnik hat dafür ein Rahmenprogramm definiert. Kursprogramme in den wichtigsten europäischen Sprachen setzen diese Vorgaben in unterschiedlichen Veranstaltungen um.

Das deutsche Kursprogramm wurde im Auftrag des Informatik-Centrum Dortmund unter der Regie des ISST erarbeitet. Ein Kurzprospekt und eine ausführliche Broschüre beschreiben die Kursinhalte und Voraussetzungen zum Erwerb des „European Professional Software Engineer“. Informationen können vom ISST angefordert werden.

Renommierte Spezialisten aus Wirtschaft und Wissenschaft konnten als Dozenten gewonnen werden. Mitarbeiter des ISST bieten Seminare zu folgenden Themen an:

- Systematisches Management von Vorgängen
- Grundlagen der Informationsmodellierung
- Integrierte Software-Infrastrukturen
- Industrialisierung von Software Engineering und Computer Aided Software Engineering
- Software-Prozeßmanagement mit FUNSOFT-Netzen
- Anforderungsanalyse und Bewertung von Software-Produktionsumgebungen
- Software Re-Engineering.

3 Anhang

3.1 Veröffentlichungen

Monographien

[Web92a] Weber, H.: Die Software-Krise und ihre Macher. Springer-Verlag Berlin Heidelberg New York, 1992.

Zeitschriften-/Buchartikel

[ADH+] Adomeit, R., Deiters, W., Holtkamp, B., Rockwell, R., Schülke, F., Weber, H.: Software Engineering Environments and Software Factories. In: Marciniak, J.J. (ed.): Encyclopedia of Software Engineering. John Wiley and Sons, New York, (in press).

[BCF+92a] Bretthauer, H., Christaller, T., Friedrich, H., Goerigk, W., Heicking, W., Hoffmann, U., Hovekamp, D., Knutzen, H., Kopp, J., Kriegel, E.U., Mohr, I., Rosenmüller, R., Simon, F.: Das Verbundvorhaben APPLY: Ein modernes und bedarfsgerechtes Lisp. In: KI, 2:5-54, Juni 1992.

[Bud92c] Budach, L.: Recursive VLSI-Design and Fractals. North-Holland Amsterdam, 1992.

[DGW93] Deiters, W., Gruhn, V., Weber, H.: Software Process Evolution in MELMAC. In: Cooke, D. (ed.): The Impact of CASE on the Software Development Life Cycle, World Scientific Series in Computer Science. 1993. Beitrag 1992 eingereicht und zur Veröffentlichung akzeptiert.

[HEV92] Holtkamp, B., Eliassen, F., Veijalainen, J.: S-Transactions. In: Elmagarmid, A. (ed.): Database Transaction Models for Advanced Applications. Morgan Kaufmann, 1992.

[Hol92a] Holtkamp, B.: EUREKA Software Factory CoRe - A Conceptual Reference Model for Software Factories. ESF Brochure, November 1992.

[Ho193] Holtkamp, B.: Software Engineering Environments and Software Factories. In: Encyclopedia of Software Engineering. John Wiley & Sons, 1993.

[HW92] Holtkamp, B., Weber, H.: Object Management Machines. In: Journal of Systems Integration, 1992.

[Web92d] Weber, H.: Einführung zum Buch: AD/Cycle - Ziele, Konzepte und Funktionen. R. Oldenburg Verlag München Wien, 1992.

Konferenzbeiträge

[ADHW92] Adomeit, R., Deiters, W., Holtkamp, B., Weber, H.: K/2r: A Kernel for ESF Software Factory Support Environment. In: Proceedings of the 1st International Conference on Systems Integration, Morristown, New Jersey, June 1992.

[DEL92] Dewal, S., Emmerich, W., Lichtinghagen, K.: A Decision Support Method for the Selection of Object Management Systems. 1st International Conference on Systems Integration, Morristown, N.J., June 1992.

[Dew92] Dewal, S.: A Methodology for Requirements Analysis and Evaluation of SDEs. 4th Conference on Advanced Information Systems Engineering, Manchester, U.K, May 1992.

[DFG593] Deiters, W., Fehling, R., Gruhn, V., Schäfer, W.: Schlüsselkomponenten der Forschungsinstanz der EUREKA Software Factory. In: Proceedings of ONLINE93, Hamburg, 1993. Beitrag 1992 eingereicht, Veröffentlichung akzeptiert.

[Oab92b] Gabriel, P.: The Object-Based Specification Language Pi: Concepts, Syntax, and Semantics. In: COMPASS Proceedings, Springer Lecture Notes in Computer Science. 8th ADT - 3rd COMPASS Workshop 1991, 1992.

[HAD93] Holtkamp, B., Adomeit, R., Deiters, W.: Work Coordination in Kernel/2r. ICO´93 Conference (angenommen), 1993.

[Hol92c] Holtkamp, B.: Kernel/2r: A Kernel for ESF Software Factory Support Environment. 2nd International Conference on Systems Integration, Morristown, N.Y., June 1992.

[Hol92f] Holtkamp, B.: MUSE Interoperation Support. ESF Seminar, Berlin, November 1992.

[Ho192g] Holtkamp, B.: Process Engine Interoperation. Workshop on Process-sensitive Software Engineering Environment Architectures, Boulder, CO, September 1992.

[Kri92c] Kriegel, U.E.: Memory Management in APPLY. International Workshop on Memory Management, Saint Malo, France, 1992.

[Mes92a] Messer, B.: Betrachtung zirkulärer Strukturen. In: 3. GI-Workshop Reflexion, Introspektion und Meta-Level- Architekturen. Gesellschaft für Informatik, März 1992.

[Mes92b] Messer, B.: Mykado - Eine Programmatik für ein Mykologisches Multimedia-Ausbildungs- und Dokumentationssystem. In: Graphische Datenverarbeitung als Werkzeug der Ausbildung und Weiterbildung. FA 4.1, Gesellschaft für Informatik, Darmstadt, Mai 1992.

[Web92b] Weber, H.: Das Projekt Eureka Software Factory: Status, Ausblick, Marktwirkung. Hauptvortrag, CASE-World 92, Hamburg, November 1992.

[Web92i] Weber, H.: Industrial Application of Software Engineering as the Victim of Programming. 5th International Conference on Industrial and Engineering Application and Expert Systems, June 9-12, 1992.

[Web92j] Weber, H.: K/2R: A Kernel for the ESF Software Factory Support Environment. 2nd International Conference on Systems Integration, June 1992.

[Web92m] Weber, H.: Offene Systeme - Fragen und Antworten in der Anwendungsentwicklung. 3. Kölner Software Tage, September 1992.

[Wik92] Wikarski, D.: Markovian Object Nets - the Conceptual Model. In: Burkhard, H.D., Czaja, L., Starke, P.H. (Hrsg.): Proc. Workshop on Concurrency, Specification and Programming (CSP 92), Seminarberichte. Humboldt-Universität Berlin, November 1992.

[XF93] Xuequn, W., Finkhoff, G.: InGRAPH: Graphical Interface for a Fully Object-Oriented Database System. ACM SAC´93 Conference (angenommen), 1993.

Sonstige wichtige Vorträge

[Ada92] Adametz, H.: Pi - An Object-Based Specification Language for Modular Concurrent Distributed Systems. COMPASS Subgroup Workshop on Specification of Concurrent Distributed Systems, Workshop lecture, March 1992.

[Dei92b] Deiters, W.: Support for Interworking in Kernel/2r. ESF Seminar 1992, November 1992.

[Gab92a] Gabriel, P.: Die Spezifikationssprache Pi: Konzepte, Syntax und Semantik. Forschungskolloquium TFS, FB Informatik, Technische Universität Berlin, März 1992.

[Got92a] Gottschick, J.: Kernel/2r. Kolloquium Technische Universität Berlin, Lehrstuhl Prof. Mahr, Mai 1992.

[Got92c] Gottschick, J.: Systemverwaltung. German UNIX User Group, Berlin, April 1992.

[Web92c] Weber, H.: Die Software-Fabrik: Von der Metapher zum operationalen Konzept. Automationsreferenten-Tagung, SNI, Dresden, Juni 1992.

[Web92e] Weber, H.: Empfehlungen für die Gestaltung von Langzeitprojekten und für die Entwicklung von Langzeitprodukten. Finanzministerium Bremen, Januar 1992.

[Web92f] Weber, H.: Forum der Eureka Software Factory. Tutorium, CASE World Konferenz: Software Produktion - Eine notwendige Unternehmensaufgabe, Congress Centrum Hamburg, November 1992.

[Web92k] Weber, H.: Können sich deutsche Hochschullehrer die EG-Forschungsförderungen leisten? Vortrag vor den Kanzlern der deutschen Universitäten, EG Brüssel, Juni 1992.

[Web92l] Weber, H.: Möglichkeiten und Grenzen für gemeinschaftliche Software-Entwicklung in Industrie, Verwaltung und Forschung. ADV- Arbeitsgemeinschaft der Prüfdienste nach § 274 SGB V, Rotenburg a. d. Fulda, Januar 1992.

[Web92n] Weber, H.: Perspektiven für die Software-Industrien im westfälischen Ruhrgebiet im allgemeinen Strukturwandel der Informations- und Kommunikationstechnik. IHK Dortmund, Industrie-Ausschuß, Juli 1992.

[Web92o] Weber, H.: Softech-nrw. Auftaktveranstaltung, Ministerium für Wirtschaft, Mittelstand und Technologie des Landes Nordrhein-Westfalen, Oktober 1992.

[Web92p] Weber, H.: Software Engineering: State of the Art - State of the Practice. Deutsche Gesellschaft für Luft- und Raumfahrt, Köln, September 1992.

[Web92q] Weber, H.: Software Engineering: Was ist das in 1992? Kraftwerks- und Simulatorgesellschaft, Essen, August 1992.

[Web92r] Weber, H.: Software re-engineering fundamentals. ISST Berlin, September 1992.

Technische Berichte

[ABF+92] Adametz, H., Budach, L., Freitag, U., Gabriel, P., Richter, M., Wikarski, D.: Motivation, Concepts and Tools of the Specification Language Pi: A Short Introduction. Interner Bericht, Dezember 1992.

[Ada93a] Adametz, H.: Context Conditions of the Pi-Language. Interner Bericht, April 1993.

[Ada93b] Adametz, H.: Konzept eines Kontextcheckers für die Pi-Umgebung. Interner Bericht, Februar 1993.

[BCF+92b] Bretthauer, H, Christaller, T., Friedrich, F., Goerigk, W., Heicking, W., Hoffmann, U., Hovekamp, D., Knutzen, H., Kopp, J., Kriegel, E.U., Mohr, I., Rosenmüller, R., Simon, F.: Das Verbundvorhaben APPLY: Ein modernes und bedarfsgerechtes Lisp. APPLY-Arbeitspapier APPLY/GMD/XIII/5, CAU/ GMD/ISST/VW-GEDAS, Sankt Augustin, Juli 1992.

[BCF+92c] Bretthauer, H, Christaller, T., Friedrich, F., Goerigk, W., Heicking, W., Hoffmann, U., Hovekamp, D., Knutzen, H., Kopp, J., Kriegel, E.U., Mohr, I., Rosenmüller, R., Simon, F.: APPLY: A modern and practical Lisp. APPLY-Arbeitspapier APPLY/GMD/XIII/6, CAU/GMD/ISST/VW-GEDAS, Sankt Augustin, September 1992.

[BCF+92d] Bretthauer, H, Christaller, T., Friedrich, F., Goerigk, W., Heicking, W., Hoffmann, U., Hovekamp, D., Knutzen, H., Kopp, J., Kriegel, E.U., Mohr, I., Rosenmüller, R., Simon, F.: Halbjahresbericht des Verbundvorhabens APPLY. 1.1.-30.6.1992. APPLY-Arbeitspapier APPLY/GMD/XIII/7, CAU/GMD/ISST/VW-GEDAS, Sankt Augustin, August 1992.

[Bud92a] Budach, L.: Compilation of the Type View in Pi. ISST-Bericht 1/92, Berlin 1992.

[Bud92b] Budach, L.: Programming in Pi - Primer and Method Guide. Interner Bericht, Berlin, Dezember 1992.

[Bud92d] Budach, L.: The Provably Intractable. Dagstuhl-Seminar-Bericht, 31:7, Februar 1992.

[DG92] Deiters, W., Gruhn, V.: The FUNSOFT Net Approach to Software Process Management. ISST-Bericht 2/92, ISST, Berlin/Dortmund, November 1992.

[FG93] Freitag, U., Gabriel, P.: Dokumentationsrichtlinien. Interner Bericht, 1993.

[FHK+92] Friedrich, H., Heicking, W., Kriegel, E.U., Mohr, I., Rosenmüller, R.: TAIL: Eine getypte Implementationssprache für APPLY. APPLY-Arbeitspapier APPLY/ ISST/IV.5/1, ISST Berlin, Juni 1992.

[FR92] Friedrich, H., Rosenmüller, R.: Die maschinennahe Zwischensprache (MZS). APPLY-Arbeitspapier APPLY/ISST/IV.5/2, ISST, Berlin, Mai 1992.

[Fre93a] Freitag, U.: Die Fensterhierarchie von PiSA. Interner Bericht, März 1993.

[Fre93b] Freitag, U.: Grobkonzept für den ASCII-Import in PISA. Interner Bericht, Februar 1993.

[FRW93] Freitag, U., Richter, M., Wikarski, D.: A Coarse State Transition Model for Pi-Components in PISA. Interner Bericht, ISST Berlin, Januar 1993.

[Gab93] Gabriel, P.: The Normalized Grammar of Pi 92 for Configurations. Interner Bericht, März 1993.

[GM93] Gabriel, P., Manhart, S.: Konzept für einen nicht-modalen Fehlermelder in PISA. Interner Bericht, März 1993.

[HR92] Heicking, W., Rosenmüller, R.: Machine Description. APPLY-Arbeitspapier APPLY/ISST/IV.5/3, ISST, Berlin, November 1992.

[KK92] Kind, A., Knutzen, H.: Typinferenz in APPLY. APPLY-Arbeitspapier APPLY/ CAU&ISST/II.1/1, ISST Berlin, Christian-Albrechts-Universität zu Kiel, November 1992.

[KM92] Kriegel, U.E., Mohr, I.: Thesen zur Objektrepräsentation in APPLY. Internes APPLY-Papier, 1992.

[KMB92] Koch, D., Messer, B., Barthel, B.: Vorgehensweise beim Aufbau einer effizienten Verwaltung mit Hilfe einer Kommunikations- und EDV-gestützten Systemlösung für den Landkreis Belzig unter Berücksichtigung des konzipierten Neukreises. Studie ISST Berlin, Dezember 1992.

[Kri92a] Kriegel, U.E.: Automatische Verwaltung dynamisch erzeugter Objekte in partiell kooperativen Umgebungen. APPLY-Arbeitspapier APPLY/ISST/VI.1/1, ISST Berlin, Januar 1992.

[Kri92b] Kriegel, U.E.: Memory Management in APPLY. APPLY-Arbeitspapier APPLY/ ISST/VI.l/3, ISST Berlin, April 1992.

[LF93] Lange, S., Freitag, U.: Die Realisierung der Komponente FILESYSTEM unter UNIX. Interner Bericht, März 1993.

[Lim92] Liman, A.: Verteilte Modellierung und Ausführung von Software-Prozeßmodellen. Thesen zur Dissertation, Universität Dortmund, Lehrstuhl für Softwaretechnologie, August 1992.

[PP93] Pi-Projekt-Grupe: Pl Software engineering Assistant. Benutzerhandbuch, ISST Berlin, März 1993.

[Web92t] Weber, H.: Die Auswahl von Werkzeugen und Umgebungen und die Einführung von CASE-Technologien. ISST-Bericht 5/92, Berlin 1992

[Wit92] Witschurke, R.: Wiederverwendung in der Informationsverarbeitung. ISST-Bericht 4/92, Berlin 1992.

3.2 Veranstaltungen und Präsentationen

Veranstaltungen

- Seminar „Vorgangsmanagement mit FUNSOFT Netzen und MELMAC“ für das Projekt FISCUS. Dortmund, 19.-21. August 1992
- Seminar „Vorgangsmanagement mit FUNSOFT Netzen und MELMAC“ für die KSG. Dortmund, 20.-23. Oktober 1992

Präsentationen

[Dei92c] Deiters, W.: MELMAC. Juni 1992. IHK Dortmund.

[GN92] Gottschick, J., Nentwig, L.: Systemadministrations-Zentrum: Status, Anforderungen und Lösungsmöglichkeiten. Dezember 1992. Workshop beim LIT.

[Got92b] Gottschick, J.: SINUS: Systemmanagement in großen Netzen. Juli 1992. Workshop beim LIT.

[Hol92b] Holtkamp, B.: ISIS - Integrierte Software-Infrasstrukturen. September 1992. Besuch von Vertretern der Senatsverwaltung für Wissenschaft im ISST.

[Hol92d] Holtkamp, B.: Kernel/2r. Juni 1992. BMFT und Presse.

[Hol92e] Holtkamp, B.: Kernel/2r Support for Systems Management. Juli 1992. Workshop beim LIT.

[Hol92h] Holtkamp, B.: Software-Bus-Konzepte. April 1992. Team SEU des BMF-Projekts „FISCUS“.

[Hol92i] Holtkamp, B.: Software-Fabrik-Konzepte. Mai 1992. Leitung der Software-Entwicklung bei Siemens.

[Hol92j] Holtkamp, B.: Werkzeuge zur Vorgehensmodellierung. Juli 1992. IHK Dortmund.

[Hol92k] Holtkamp, B.: System Kernel/2r. 25.-27. November 1992. ESF-Seminar, Berlin.

3.3 Dissertationen und Diplomarbeiten

Wolfgang Deiters

- A View based Approach to Software Process Management.
  Dissertation an der TU Berlin, FB Informatik, Dezember 1992

Sanjay Dewal

- A Methodology for Requirements Analysis and Evaluation of Software Developments Environments.
  Dissertation an der Universität Dortmund, FB Informatik, Januar 1992

Burkhard Messer

- Fachsprachliche Entwicklung einer Theorie der Programme.
  Dissertation an der TU Berlin, FB Informatik, Dezember 1992

A. Liman

- Verteilte Modellierung und Ausführung von Software-Prozeßmodellen.
  Diplomarbeit, Universität Dortmund, FB Informatik, Lehrstuhl für Software-Technologie, August 1992

3.4 Mitarbeit in Gremien, Konferenz- und Programmkomitees

Prof. Herbert Weber

- *Landesbeauftragter* des Landes Nordrhein-Westfalen für Informationstechnik
- ESF Control Board Speaker
- *Associate Editor,* IEEE Trans. on Knowledge and Data Engineering
- *Associate Editor,* WEE Trans. on Software Engineering
- *Program Chair,* ACM SIGSOFT´92, Fifth Symposium on Software Development Environments; 9.-11. Dezember 1992, Tyson´s Corner, Virginia, USA
- *European Program Chair,* The First International Conference on Systems Integration; 15.-18. Juni 1992, Morristown, New Jersey, USA
- *European Coordinator,* Eighth International Conference on Data Engineering; 3.-7. Februar 1992, Phoenix, Arizona, USA
- *Member of Advisory Council,* Fifth International Workshop on Computer-Aided Software Engineering; 6.-10. Juli 1992, Montreal, Quebec, Canada

Dr. Horst Friedrich:

- Beteiligung an der europäischen Arbeitsgruppe zur LISP-Standardisierung

Dr. Bernhard Holtkamp

- Teilnahme am Treffen der ECMA/TC33/TGRM, Genf
- *Program Committee Member,* The First International Conference on Systems Integration; 15.-18. Juni 1992, Morristown, New Jersey, USA

3.5 Vorlesungstätigkeit an Hochschulen

Prof. Lothar Budach

- Vorlesung „Informatik 1“, Universität Potsdam, WS 1992/93

Jürgen Oheim

- Vorlesung „Grundlagen der Informationsmodellierung“, TU Berlin, WS 92/93

Prof. Herbert Weber

- Vorlesung „Software Entwurf“, TU Berlin, WS 92/93