Aus: Das ist Informatik, Jörg Desel (Hrsg.), Springer Verlag, 2001.

# Softwaresystemtechnik − eine Informatik−Ingenieurdisziplin

*Prof. Siegfried Wendt, April 2000* − http://www.hpi.uni−potsdam.de/

## Informatik zwischen Grundlagenwissenschaft und Ingenieurwissenschaft

Ist Maschinenbau eine Ingenieurwissenschaft? Ist Physik eine Grundlagenwissenschaft? Hierüber gibt es keine Debatten, weil bezüglich der Antworten längst ein stabiler Konsens besteht. Die Frage dagegen, ob Informatik eine Ingenieurwissenschaft sei, wird immer noch diskutiert. Muss sie überhaupt entschieden werden, das heißt, muss denn jede Disziplin eindeutig auf die eine oder auf die andere Seite festgelegt werden? Im Vergleich zur Physik und zum Maschinenbau ist die Informatik noch in den Jugendjahren, und man muss ihr noch eine längere Reifezeit zugestehen. Man wird das Erwachsensein daran erkennen, dass die Debatte über die Frage, für welche Inhalte man zuständig sei und wie man die Gegenstände der Disziplin kommunizieren solle, zu Ende gekommen ist.

Wenn man Informatik als das nimmt, was der Fakultätentag als Fächerkanon anerkennt, stellt man fest, dass die Informatik in vielerlei Hinsicht "sowohl als auch" sein will. Es werden nämlich keine Prioritäten gesetzt: Weder in der Frage, ob der Diplominformatiker mehr in Richtung eines Wissenschaftlers oder mehr in Richtung eines Projektleiters für die Softwareindustrie ausgebildet sein soll, noch in der Frage, ob er in der Industrie vorwiegend mit der Erstellung von Programmen oder mit der Betreuung von Entwicklungsprojekten befasst sein soll, gibt es einen Konsens bezüglich der Antwort. Die Informatik als Oberbegriff erfordert es tatsächlich nicht, dass auf diese Fragen eindeutige Antworten gegeben werden. Da es aber unmöglich ist, die Studenten für alle diese unterschiedlichen Aufgaben in gleichem Maße optimal auszubilden, müssen entsprechende Ausbildungsalternativen angeboten werden. Inzwischen gibt es genügend Symptome für Ausbildungsdefizite, die es nahe legen, alternative Studiengänge anzubieten. In der Frühzeit der Informatik, also vor über 30 Jahren gab es für solche Unterteilungen selbstverständlich noch keine Substanz. Die damalige Entscheidung, die Informatikausbildung formal wie eine Mathematikausbildung zu strukturieren und nicht wie eine Ingenieurausbildung, erkennt man auch heute noch ganz eindeutig daran, dass die Rahmenprüfungsordnung und die Form der Diplomzeugnisse zu den entsprechenden Dokumenten der Mathematik eine verblüffende Ähnlichkeit aufweisen, während sie sich von den entsprechenden Dokumenten der Ingenieurwissenschaften grundsätzlich unterscheiden. Die Tatsache, dass man den Begriff "Angewandte Informatik" eingeführt hat in Entsprechung zu Begriffen wie Angewandte Mathematik oder Angewandte Physik, deutet darauf hin, dass sich die Informatik nicht primär als Ingenieurwissenschaft versteht. Denn in den Ingenieurwissenschaften gibt es keine Verwendung des Attributs "angewandt", weil dies ein Pleonasmus der Art "weißer Schimmel" wäre.

Die Tatsache, dass der überwiegende Teil der Diplominformatiker als Programmentwickler eingesetzt wird, der den größten Teil seiner Zeit damit zubringt, programmiersprachliche Texte zu konstruieren, wird im allgemeinen unkommentiert hingenommen. Es muss aber gefragt werden, ob dies das Ziel der Ausbildung zum Diplominformatiker gewesen sei. Es wäre sicher nichts dagegen einzuwenden, wenn ein Diplominformatiker zu höchstens 10 % seiner Zeit mit der Entwicklung programmiersprachlicher Texte befasst wäre; ernsthafte Einwände müssen jedoch erhoben werden, wenn dieser Zeitanteil die 30 %− Marke erreicht oder gar überschreitet. In der Analogie zu den traditionellen Ingenieurwissenschaften sollte man nämlich das Erstellen programmiersprachlicher Texte der Fertigung technischer Bauteile mit einer Werkzeugmaschine − einer Drehbank oder einer Fräsmaschine − gleichstellen. An diesen Maschinen arbeiten keine Ingenieure, sondern speziell hierfür ausgebildete Facharbeiter. Die Frage nach dem Informatikfacharbeiter ist zur Zeit noch nicht befriedigend beantwortet.

> Der Ingenieur

# Typische Merkmale von Ingenieurwissenschaften

Ingenieurwissenschaften werden erforderlich, wenn das Basteln technischer Produkte nicht mehr zu befriedigenden Ergebnissen führt. Durch die Ingenieurwissenschaften wird die Entwicklung und die Herstellung technischer Produkte auf eine solide wissenschaftliche Basis gestellt. Diese ist gekennzeichnet durch eine angemessene Begriffswelt, dazu passende Darstellungsformen und Methoden für das systematische Gewinnen von Ergebnissen. Die Ingenieurwissenschaft läuft zwangsläufig immer der Bastelei hinterher: Die Thermodynamik kam erst nach der Dampfmaschine.

Wer nicht weiter darüber nachdenkt, könnte voreilig zu dem Schluss kommen, Ingenieurdisziplin sei im Grunde angewandte Physik und angewandte Mathematik. Zweifellos erledigt ein Ingenieur in den traditionellen Disziplinen immer wieder Aufgaben der angewandten Physik und der angewandten Mathematik. Dennoch wäre es nicht angemessen, einen sogenannten angewandten Physiker oder einen angewandten Mathematiker als Ingenieur zu bezeichnen. Das wesentliche Kennzeichen des Problembereichs der Ingenieure ist die Befassung mit komplexen Systemen. Und dabei geht es gerade um die nicht mathematisch behandelbaren Aspekte dieser Systeme, also nicht um Erkenntnisse und Entscheidungen, die man berechnen kann, sondern um Entscheidungen, die man nur mehr oder weniger zweckmäßig oder unzweckmäßig fällen kann. Es geht dabei nicht immer nur um die Konstruktion von neuen Systemen, sondern auch um Fragen ihrer evolutionären Veränderung oder ihrer Kopplung mit anderen Systemen.

Die Lehrinhalte eines ingenieurwissenschaftlichen Studienganges müssen immer von der Frage her gefunden werden, was die Absolventen gelernt haben sollten, damit sie den Aufgaben, denen sie in der Industrie begegnen, gewachsen sind. Die Frage, ob es sich dabei auch um Lehrinhalte handelt, die für den Hochschullehrer wissenschaftlich interessant sind, sollte immer erst an zweiter Stelle gestellt werden.

Die Komplexität der Systeme bringt es zwangsläufig mit sich, dass die Systeme nur arbeitsteilig erstellt und verändert werden können. Da der Ingenieur eines Tages durchaus in der Lage sein sollte, die Verantwortung für den gesamten Prozess der Systemgestaltung und Pflege zu übernehmen, muss er primär im Hinblick auf diesen arbeitsteiligen Prozess ausgebildet werden und nicht im Hinblick auf die Optimierung einzelner Schritte dieses Prozesses. Der Ingenieur kann immer davon ausgehen, dass er Spezialisten für Spezialaufgaben zur Verfügung haben wird. Er muss zuallererst einmal als "Kommunikateur technischer Inhalte" ausgebildet werden, der seine Planungen unmissverständlich weitergeben kann und der mit seinen Spezialisten optimal kommunizieren kann. Es gehört unbedingt zum Weltbild des Ingenieurs, dass ihm immer der Unterschied zwischen einem technischen Produkt und dem Ergebnis einer Bastelei, und sei sie auch noch so genial, bewusst bleibt. Labormuster und technische Produkte wird ein Ingenieur nie verwechseln.

> Software Engineering

# Der schillernde Begriff "Software Engineering"

Der Begriff Software Engineering wurde vor rund 30 Jahren geprägt, also um die Zeit, als in Deutschland die Fachbereiche Informatik eingerichtet wurden. Damals war man sich gerade bewusst geworden, dass es einen Unterschied zwischen "Progamming in the small" und "Programming in the large" gibt. Durch die Einführung des Begriffes Software Engineering sollte den Informatikern bewusst gemacht werden, dass sie sich die traditionellen Ingenieurdisziplinen zum Vorbild nehmen können, wenn es um die Frage geht, wie sie die Qualität und die Produktivität auf dem Gebiet der Softwareproduktion verbessern könnten. Allerdings gibt es nicht nur außerhalb der Ingenieurwissenschaften, sondern sogar innerhalb der Ingenieurwissenschaften recht unterschiedliche Vorstellungen bezüglich der Frage, was denn das Typische an einer ingenieurmäßigen Vorgehensweise sei. Deshalb gibt es noch keine übereinstimmende Antwort auf die Frage, worauf man denn im Software Engineering den Schwerpunkt legen soll. Die Mehrheit ist sich zumindest darin einig, dass es ein wesentlicher Problemkreis innerhalb des Software Engineering sei, wie man von einem formal spezifizierten System zu einer verifizierten Implementierung kommt.

Bezüglich der Frage, ob Software Engineering als Teilgebiet innerhalb eines Studienganges Informatik gelehrt werden solle, oder ob Software Engineering ein eigener Studiengang sein müsse, gibt es noch keinen Konsens. David Parnas hat in dieser Frage sehr deutlich Stellung bezogen, die er in seinem Aufsatz "Software Engineering Programmes are not Computer Science Programmes" [1] zum Ausdruck gebracht hat. Es wäre in diesem Kontext wohl unangemessen, Computer Science mit Informatik gleichzusetzen. Computer Science sollte wohl eher als eine Richtung innerhalb der Informatik betrachtet werden. Der Autor ist sich mit David Parnas einig in der Überzeugung, dass sich das Ausbildungsprogramm für einen Informatik−Grundlagenwissenschaftler grundlegend vom Ausbildungsprogramm für einen Informatik−Ingenieur unterscheiden muss.

> Kommunikationsprobleme

## Symptome eines grundsätzlichen Defizits

Man stelle sich zwei Universitätsabsolventen vor, von denen der eine Diplomingenieur im Maschinenbau und der andere Diplominformatiker geworden ist. Der Maschinenbauer tritt seine erste Arbeitsstelle bei der Flugzeugfirma Boeing an, der Diplominformatiker bei der Softwarefirma SAP. Der Jungingenieur bei der Firma Boeing soll mithelfen, eine Schwachstelle im Antrieb des Seitenruders konstruktiv zu verbessern; der Junginformatiker bei SAP soll mithelfen, ein CAD−System einer Zulieferfirma an das System R/3 der Firma SAP anzukoppeln. Wie lange wird es wohl dauern, bis jeder der beiden Jungakademiker jeweils genau weiß, wie die Systemkomponenten, um die er sich kümmern soll, ins Gesamtsystem eingebettet sind und welche konstruktiven Alternativen er hat, sein Ziel zu erreichen? Der Maschinenbauer hat keine Mühe, sich den erforderlichen Überblick innerhalb des ersten Monats zu verschaffen, wogegen der Informatiker auch noch nach einem halben Jahr das ungute Gefühl hat, dass er möglicherweise über wichtige Zusammenhänge immer noch nicht Bescheid weiß.

Nun betrachten wir ein zweites Szenario. Ein Professor der Kraftfahrzeugtechnik besucht eine Automobilfirma, und ein Professor für Software Engineering besucht eine Softwarefirma. Die Ingenieure in der Automobilfirma werden den Besucher wie ihresgleichen behandeln, also wie einen fachkundigen Kollegen, und sie werden ihn dementsprechend über konstruktive Details anhand von Konstruktionsplänen und anhand von Labormustern informieren. Die Informatiker in der Softwarefirma dagegen werden ihren Besucher nicht wie einen Fachkollegen behandeln, sondern wie einen Journalisten oder Politiker. Sie werden ihm eine möglichst beeindruckende Demonstration der reichen Funktionalität ihrer Systeme bieten, über konstruktive Details im Inneren dieser Systeme werden sie aber in den seltensten Fällen ein Wort verlieren. Hätten die Ingenieure der Automobilfirma sich ähnlich verhalten, so hätten sie ihren Besucher zu einer Probefahrt im neuen Auto eingeladen.

Interessanterweise werden nicht nur Besucher in Softwarefirmen wie Journalisten oder Politiker mit Demonstrationen von Funktionalität abgespeist, sondern dies ist sogar der übliche Kommunikationsstil zwischen Entwicklern und ihren Projektleitern. Die Berichtspflicht zwischen Untergebenen und Vorgesetzten wird nicht dadurch erfüllt, dass konstruktive Details dargestellt werden, sondern immer nur dadurch, dass Funktionalität vorgeführt wird. Es wird also stillschweigend angenommen, dass alles in Ordnung sei, solange man beweisen könne, dass man das System zum Laufen gekriegt hat.

In beiden Szenarien äußert sich das gleiche grundsätzliche Defizit im heutigen Software Engineering: Man hat nicht gelernt, wie man angemessen von Ingenieur zu Ingenieur über die wesentlichen Inhalte der Ingenieursarbeit kommunizieren soll.

> Der Studiengang

# Der Studiengang "Softwaresystemtechnik" am HPI in Potsdam

Hätte der Studiengang die Bezeichnung Software Engineering bekommen, hätte sich niemand aufgefordert gefühlt, nach den Inhalten zu fragen, denn weil Software Engineering ein schon lange eingeführter Begriff ist, glaubt jeder zu wissen, was er sich darunter vorzustellen habe. Selbstverständlich fällt der Begriff Softwaresystemtechnik unter den Begriff Software Engineering, denn es handelt sich ja um einen Ingenieurstudiengang, der der Informatik zuzuordnen ist. Die Besonderheit liegt in der bewussten Schwerpunktsetzung auf die Frage, wie das Wissen über komplexe Softwaresysteme optimal kommuniziert werden kann.

Die Komplexität der heutigen Softwaresysteme ist eine zwangsläufige Konsequenz der seit Jahrzehnten exponentiell gestiegenen und immer noch exponentiell weiter steigenden Leistungsfähigkeit der Hardware, sowohl was die Verarbeitungsgeschwindigkeit als auch was die Speicherkapazität angeht. Der Entwicklungsbedarf für Software und der Umfang der Softwaresysteme wird in den kommenden Jahrzehnten noch in ungeahntem Maße ansteigen.

Die wissenschaftliche Behandlung der Methodik des Baus von Softwarekomponenten ist schon vor langer Zeit und an sehr vielen Orten in Angriff genommen worden und hat deshalb inzwischen einen recht hohen Reifegrad erreicht. Auf diesem Gebiet steht also die Lehre bereits auf einem breiten und soliden Fundament von Erkenntnissen. Es ist selbstverständlich, dass diese Erkenntnisse auch im Studiengang Softwaresystemtechnik vermittelt werden. Der Komponentenbereich zeichnet sich dadurch aus, dass hier formale Spezifikation möglich ist, die als Ausgangspunkt für den formal unterstützten Lösungsprozess genommen werden kann. Das Ergebnis des Lösungsprozesses kann dann formal daraufhin überprüft werden, ob die Spezifikation korrekt erfüllt wurde. Der Übergang von einer formalen Spezifikation zu einer verifizierbaren Lösung ist aber nicht das typische Kennzeichen der Ingenieursarbeit, sondern fällt eher in den Bereich der angewandten Mathematik.

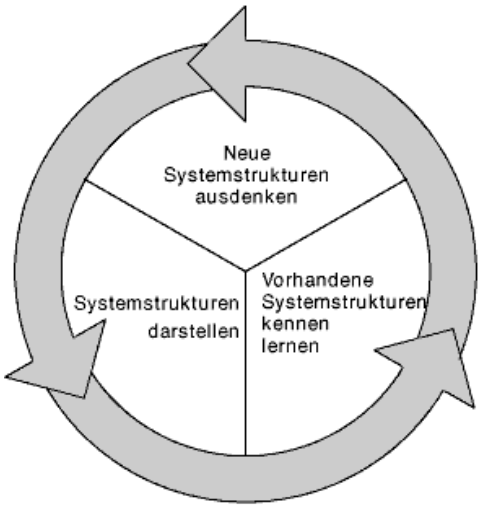
Das Typische der Ingenieursarbeit wird nicht durch den Komponentenbau bestimmt, sondern durch die Systemkonzeption. Hierfür gibt es keine formale Spezifikation und demzufolge keine formal unterstützbaren Lösungswege. Eine zweckmäßige Systemkonzeption kann dem Ingenieur nur gelingen, wenn er auf zwei unterschiedlichen Bereichen über Erfahrungen verfügt: Zum einen muss er klare Vorstellungen über die funktionelle Vielfalt machbarer Komponenten haben, und zum anderen muss er viele aktuell in der Vergangenheit realisierte Systeme kennen gelernt haben.

In den traditionellen Ingenieurwissenschaften wie Maschinenbau und Elektrotechnik gibt es selbstverständlich eine Fülle von Katalogen, in denen eine Vielfalt von Komponenten in funktioneller Klassifikation zusammengestellt sind. Für die vorliegende Betrachtung sind dabei nicht die Kataloge für die elementaren Komponenten wie Schrauben, Zahnräder, Widerstände oder Transistoren interessant, sondern die Kataloge für Komponenten, die bereits aus elementaren Bauteilen aufgebaut wurden. Man denke an Getriebe, Pumpen, Verstärker oder Antennen. Entsprechend kann sich der Ingenieur der Softwaresystemtechnik auf Kataloge für Algorithmen stützen. Die Ausbildung der Ingenieure der Softwaresystemtechnik muss dahin führen, dass der Ingenieur mühelos die Kataloge verstehen kann, um darin zielgerichtet nach geeigneten Komponenten für sein System zu suchen. Die Ausbildung hat nicht primär das Ziel, ihn in die Lage zu versetzen, neue Beiträge für die Kataloge zu schaffen.

Ein System entsteht immer als Ergebnis einer Iteration von Entwürfen. Ein Entwurf wird im Grunde immer von einem einzelnen Menschen geschaffen − dies gilt nicht nur für den Bereich der Ingenieurentwürfe, sondern überall, wo das Wort Entwurf einen Sinn hat, also beispielsweise auch für Gesetzesentwürfe oder für Entwürfe von Tapetenmustern. Die Entscheidung, ob ein Entwurf realisiert werden soll, wird nicht alleine von demjenigen gefällt, der den Entwurf geschaffen hat, sondern diese Entscheidung ist die Konsequenz einer Diskussion mehrerer Fachleute. Deshalb müssen die Entwürfe so leicht fasslich dargestellt werden, dass die zu beteiligenden Fachleute überhaupt in die Lage kommen, in der Diskussion eines vorgelegten Entwurfs konstruktive Beiträge zu leisten.

Die zentrale Rolle der Systemdarstellung ist im folgenden Bild veranschaulicht. Es sind drei Aktivitäten

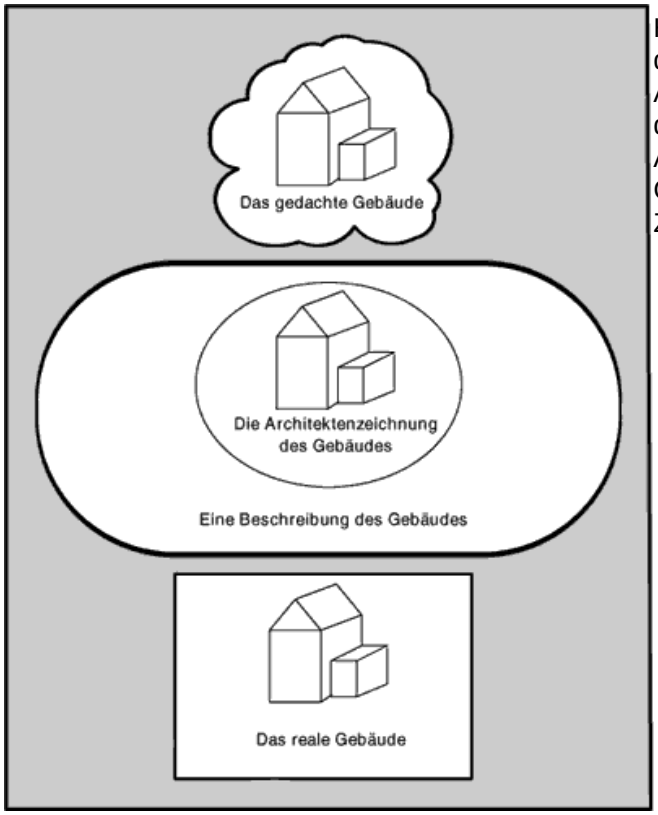
gezeigt, die zyklisch voneinander abhängen. Damit man sich ein Systemkonzept ausdenken kann, sollte man vorher schon Systemkonzepte, die in der Vergangenheit realisiert wurden, kennen gelernt haben. Damit man Systemkonzepte kennen lernen kann, müssen diese leicht fasslich dargestellt worden sein. Und damit man ein Systemkonzept darstellen kann, muss dieses zuvor von jemandem erdacht worden sein.



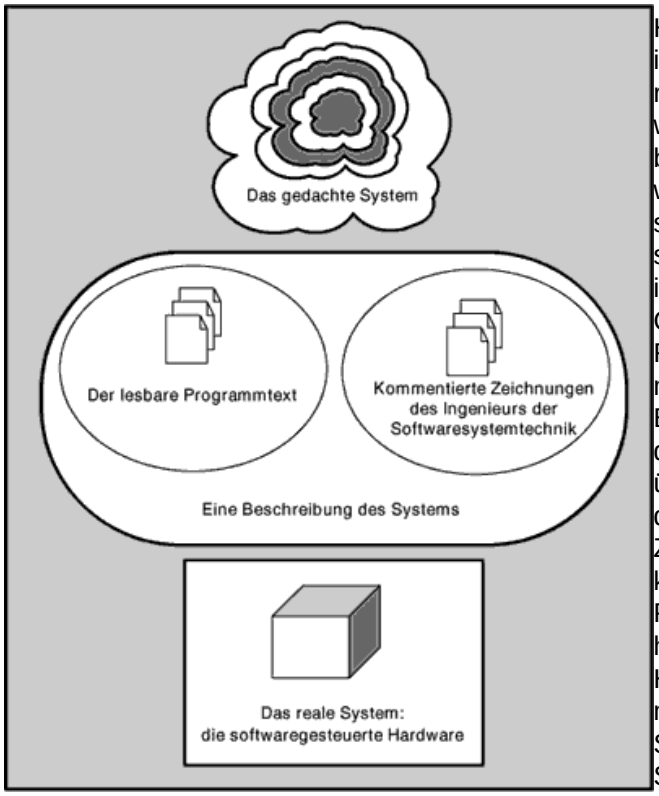

Voraussetzungen und Ziele der Ingenieurausbildung

Unter den drei aufgeführten Aktivitäten gibt es nur eine, welche die Existenzberechtigung der Ingenieure begründet. Dies ist zweifellos die Aktivität des Ausdenkens von Systemkonzepten. Daraus darf man aber nun nicht den Schluss ziehen, die Ausbildung von Ingenieuren müsste sich auf diese Aktivität konzentrieren, weil die Ingenieure besonders befähigt werden müssten, neue Ideen und neue Lösungen zu finden. Man kann niemandem beibringen, gute Einfälle zu haben. Es gilt die Analogie zur Landwirtschaft, wo man den Bauern auch nicht beibringen kann, gutes Getreide wachsen zu lassen. Man kann ihnen lediglich beibringen, was sie tun müssen, damit die Voraussetzungen für das Wachsen günstig sind. Da das eine Feld im obigen Bild also nicht direkter Gegenstand der Lehre sein kann, muss sich die Lehre auf die anderen beiden Felder konzentrieren. Man muss also den Studenten beibringen, wie Systemkonzepte darzustellen sind, und man muss ihnen sehr viele unterschiedliche Systeme aus der Vergangenheit vorstellen. Letzteres ist nur möglich, wenn man zuvor die Darstellungsmethoden gelehrt hat.

In den traditionellen Ingenieurwissenschaften wirft das Darstellen von Systemkonzepten keine besonderen Schwierigkeiten auf, denn im Bereich gegenständlicher Systeme ist das Darstellungsproblem fast trivial. Die folgenden Bilder veranschaulichen den grundsätzlichen Unterschied der Darstellungsproblematik anhand einer Gegenüberstellung der traditionellen Ingenieurwissenschaften und der Disziplin der Softwaresystemtechnik.




Hier geht es um ein Gebäude, welches zuerst nur vor dem geistigen Auge des Architekten steht. Anschließend erstellt der Architekt eine Zeichnung, damit nun jedermann sehen kann, was zuvor nur der Architekt sah. Und wenn dann schließlich das Gebäude steht, sieht es so aus wie auf der Zeichnung.




Hier dagegen geht es um ein informationstechnisches System, welches dadurch realisiert wird, dass sehr viel Softwaretext erstellt wird, der dann ein Computersystem dazu bringt, ein benutzbares Gerät mit bestimmter Funktionalität zu werden. Im Gegensatz zum Gebäudearchitekten sieht der Ingenieur der Softwaresystemtechnik vor seinem geistigen Auge keine reale Struktur. Im Bild ist dies durch die Zeichnung eines wolkenartigen Gebildes zum Ausdruck gebracht. Ganz unten ist in Form eines Würfels das reale System skizziert. Man muss sich hier die Computergehäuse, die Kabel, die Bildschirmgeräte, die Tastaturen und andere Geräte, die zum Computersystem gehören, vorstellen. Links über diesem Würfel sieht man die Dokumente, auf denen der Programmtext steht, der viele Millionen Zeilen umfassen kann. Wollte man den zu einem komplexen Softwaresystem gehörenden Programmtext auf Papier ausdrucken, erhielte man häufig Papierstapel von der Höhe mehrstöckiger Häuser. Rechts neben dem Programmtext findet man die Zeichnungen der Ingenieure der Softwaresystemtechnik, aus denen die restlichen Strukturen überhaupt erst verständlich werden. Die begrifflichen Kategorien und Konzepte, die hinter diesen Ingenieurszeichnungen der Softwaresystemtechnik stehen, sind nicht trivial und müssen den Studenten erst über mehrere Semester hinweg beigebracht werden.

In seinem Aufsatz "Defining a Discipline of Description" [2] beklagt Michael Jackson das Fehlen einer Disziplin, die das Darstellen komplexer Softwaresysteme zum Gegenstand hat. Am HPI gibt es diese Disziplin, und sie steht im Zentrum der Ausbildung. In seinem Buch "Warum ist Software so teuer?" [3] schreibt Tom DeMarco auf Seite 158: "Bei der Software−Entwicklung geht es vor allem darum, miteinander zu reden und Dinge niederzuschreiben. Diejenigen, die die Disziplin am stärksten voranbrachten, waren in aller Regel die besten Kommunikateure, nicht die besten Techniker ihres Fachgebiets." Bei der Gestaltung

des Studienganges Softwaresystemtechnik am HPI hat man daraus die Konsequenzen gezogen: Die Studenten werden zuallererst einmal zu Kommunikateuren ausgebildet, die immer genau wissen, wovon sie reden und wie sie die Inhalte darstellen müssen, damit sie von anderen verstanden werden können.

Die Analogie zwischen traditionellen Ingenieuren und Ingenieuren der Softwaresystem−technik hätte äußerst treffend durch ein Foto veranschaulicht werden können, das der Autor aufgenommen hätte, wenn er im richtigen Moment eine Kamera dabei gehabt hätte: Der Blick geht von oben auf eine Baustelle zur Errichtung eines großen Hotels. Der Bau ist noch nicht wesentlich über das Kellergeschoss hinausgewachsen. Man sieht die Bauarbeiter mit ihren farbigen Schutzhelmen − teilweise hin− und hergehend oder an verschiedenen Orten unterschiedliche Arbeiten ausführend. Einige dieser behelmten Leute sind über einen Tisch gebeugt, auf dem man Baupläne liegen sieht. Man kann vermuten, dass einer der Männer eine leitende Funktion hat, denn er scheint den anderen etwas anhand der Pläne zu erklären oder ihnen anhand der Pläne gewisse Aufgaben zuzuweisen. Es ist aber auch zu vermuten, dass dieser Mann nicht derjenige war, der die Pläne gezeichnet hat. Man nimmt es als ganz selbstverständlich an, dass die Pläne so verständlich sind, dass man keine mündlichen Kommentare des Architekten oder Bauingenieurs dazu benötigt. Pläne, die nur verständlich werden, wenn ihr Zeichner mündlich kommentiert, was er denn eigentlich mit den Symbolen und Zeichen auf dem Plan gemeint hat, wären auf dieser Baustelle völlig nutzlos.
Bei der Erstellung großer Softwaresysteme müsste es eigentlich genauso zugehen wie auf dieser Baustelle, und viele Leute, die der Informatik ferne stehen, nehmen auch an, dass dies der Fall sei. Wer jedoch die Praxis der industriellen Softwareentwicklung kennt, weiß, wie weit der Weg noch ist, bis die Analogie wirklich zutrifft. Der Studiengang Softwaresystemtechnik am HPI fördert das Vorankommen auf diesem Weg.

Die rasante Entwicklung auf dem Markt der Softwareprodukte könnte die Gestalter von Studiengängen dazu verführen, die Lehrinhalte an der schnellen Änderung der Produkte und der damit verbundenen erforderlichen Bedienungskenntnisse zu orientieren. Diese Gefahr wurde am HPI gesehen, und es wurde bewusst gegengesteuert. Die Ausbildung konzentriert sich auf das längerfristig Gültige, also auf das, was vielen Systemkonzepten oder Bedienungsstrukturen gemeinsam ist. Die Absolventen des Studienganges Softwaresystemtechnik werden während ihrer jahrzehntedauernden Berufstätigkeit zwangsläufig immer wieder neue Kenntnisse erwerben müssen. Die Qualität der Ausbildung im Studiengang Softwaresystemtechnik ist deshalb an der Frage zu messen, wie gut die Absolventen darauf vorbereitet werden, sich später autodidaktisch das neue Wissen zu erwerben. Hierfür bilden die geeigneten Darstellungsmittel eine wesentliche Grundlage.

Die Reife einer Darstellungsmethodik zeigt sich daran, dass technologische Revolutionen keine wesentlichen Änderungen der Darstellungsmittel erfordern. Der Übergang von einem Barockschloss zum Empire State Building erforderte keine grundsätzlichen Änderungen bei den Konzepten des Bauzeichnens.

## Literatur:


[1] Parnas, David Lorge: Software Engineering Programmes are not Computer Science Programmes. CRL Report 361, McMaster University, Hamilton, Canada, April 1998

[2] Jackson, Michael: Defining a Discipline of Description. IEEE Software, Sept./Oct., 1998

[3] DeMarco, Tom Warum ist Software so teuer ? Carl Hanser Verlag, München Wien, 1997