Prof. Lucas über die Einführung in die Informatik

Die Studienpläne der Telematik und der Technischen Mathematik enthalten die Lehrveranstaltungen Informatik 1 und Informatik 2 mit jeweils zwei Vorlesungsstunden im ersten bzw. zweiten Semester. Die Studienpläne sehen weiters ein Programmiermethoden Praktikum mit einer Vorlesungsstunde und zwei Übungsstunden vor. Seit vielen Jahren wird mehr oder weniger heftig über die Frage diskutiert, was denn der Inhalt dieses Blocks von Einführungsvorlesungen sein soll und ob eine Beziehung zwischen den Vorlesungen aus Informatik und dem Programmiermethoden Praktikum bestehen soll und was diese Beziehung sein könnte. Der vorliegende Artikel ist eine Beitrag zu dieser Diskussion.

Informatik

In der folgenden Auseinandersetzung wird der vor allem im deutschen Sprachraum geläufige Begriff der Informatik mit dem im englischen Sprachraum üblichen Begriff der computer science identifiziert. Was unter diesen Bezeichnungen an den Universitäten gelehrt ist inhaltlich und der Intention nach ähnlich. Die Details der Lehre sind von Universität zu Universität verschieden unabhängig von der Bezeichnung. Bei der groben Einteilung der Informatik in Teilgebiete scheint die folgende vor allem in Europa verbreitet zu sein:

THEORETISCHE INFORMATIK
ANGEWANDTE INFORMATIK
TECHNISCHE INFORMATIK

Daß sofort auf der obersten Ebene zwischen Theorie und Anwendung unterschieden wird, ist keine besonders glückliche Entscheidung. Es entsteht der — hoffentlich unbeabsichtigte — Eindruck, daß man das Eine ohne das Andere studieren kann.

Ein Studienplan, der schon im ersten Semester ein Programmier- methoden Praktikum vorsieht, verstärkt diesen Eindruck leider, da zu diesem Zeitpunkt die relevanten intellektuellen Werkzeuge ja notwendigerweise noch fehlen.

Die ACM/IEEE 2 curriculum task force fand eine weit bessere Einteilung der Informatik. Der Bericht der Arbeitsgruppe unterscheidet neun Teilgebiete, kein einziges ist nur theoretisch oder nur praktisch. (Die englischen Originaltitel der erwähnten neun Teilgebiete sind: Algorithms and Data Structures, Architecture, Artificial Intelligence and Robotics, Database and Information Retrieval, Human-Computer Communication, Numerical and Symbolic Com- putation, Operating Systems, Programming Languages, Software Methodology and Engineering.) Es ist ein ganzheitlicher Ansatz. Nach Ansicht der Arbeitsgruppe ist die Informatik *„simultaneously a mathematical, scientific, and engineering discipline"*.

Um diese Kombination zu charakterisieren, spricht die Arbeitsgruppe von drei Prozessen, die vom Praktiker anzuwenden sind: Theorie, Abstraktion und Entwurf.

Informatik und Softwareengineering

Traditionell wird Softwareengineering als Teilgebiet der Informatik gesehen und behandelt. Auch die Arbeitsgruppe der ACM/IEEE hat das so gesehen. Kürzlich hat sich David Parnas zu Wort gemeldet, der wesentlich zu unserem Gebiet beigetragen hat (z.B. der Begriff „information hiding" — also Kapselung — wird ihm zugeschrieben). In einer Arbeit mit dem Titel „Computer Science Programmes are not Software Engineering Programmes" argumentiert Parnas, daß Software Engineering nicht als Teilgebiet der Informatik behandelt werden kann, weil die ingenieursmäßige Ausbildung dabei zu kurz kommt.

David Parnas schlägt eine Trennung vor und zwar in eine Grundlagenwissenschaft Informatik und eine Ingenieursdisziplin Software Engineering.

Er meint, die Informatik sollte sich zu Software Engineering so verhalte, wie die Physik zur Elektrotechnik.

Professorenkommentar

Damit wären dann auch die Zielsetzungen besser definierbar: Ziel der Informatik wäre, neues Wissen zu suchen. Ziel des Software Engineering müßte sein, vorhandenes Wissen anzuwenden, um Systeme zu bauen.

Trotzdem zweifle ich an diesem Ansatz. Bei der schnellen Entwicklung unseres Gebietes ist eine solche Trennung vielleicht gar nicht möglich oder sinnvoll, weil das derzeitige spezielle Wissen, das im Software Engineering im Augenblick Anwendung findet, veraltet sein wird, wenn die heute Erstsemestrigen in den Beruf gehen.
Nur das Grundlagenwissen und die erworbenen intellektuellen Fertigkeiten haben eine längere Halbwertszeit.

Die mathematische Basis der Informatik

Die für die Informatik wichtigen Grundlagen liegen hauptsächlich in der formalen Logik und der abstrakten Algebra. Es hätte geholfen, wenn ich während meines Studiums mehr über diese Gebiete gelernt hätte.

Nur der Erwerb dieser Grundlagen ermöglicht die unverkrampfte Leichtigkeit des Umgangs mit den Elementen der Informatik, den Einblick in die technischen Alternativen von Pro- blemlösungen und die Fähigkeit Systeme nicht nur im Trial-and-Error-Verfahren zu hacken, sonder schöne Systeme systematisch zu entwickeln.

Gedanken zur Einführung in die Informatik

Die wesentliche Idee des Computers ist die Idee einer Maschine die Sprache interpretiert. Jede Einführung in das eigentliche Gebiet der Informatik ist daher notwendigerweise sprachorientiert. Dies bedeutet insbesondere die Einführung in ein Universum, das durch Programmiersprachen ausdrückbar ist. Notationelle Belange sind dabei von untergeordneter Bedeutung. Für eine Einführungsvorlesung muß hier eine Entscheidung getroffen werden. Grob gesprochen stehen drei semantische Modelle von Programmiersprachen zur Auswahl.

1. das funktionale Modell

2. das imperative Modell

3. das relationale Modell
(wird hier nicht besprochen).

Das funktionale Modell

Das Universum des funktionalen Modells besteht aus gewissen elementaren und zusammengesetzten Datenelementen und Funktionen über diesen Bereichen. Eine wesentliche Charakteristik funktionaler Sprachen ist, daß auch Funktionen höhere Ordnung miteingeschlossen sind. Imperativen Sprachen liegt ein Bereich von Zuständen und Zustandstransformationen zugrunde, d.h. abstrakte Maschinen. Funktionale Sprachen sind in vieler Hinsicht einfacher und abstrakter als imperative Sprachen.

Funktionale Sprachen haben nützliche mathematische Eigenschaften, die in Korrektheitsargumenten und Programmtransformationen Verwendung finden. Das ist keine Konsequenz des funktionalen Modells als solchem, sondern Funktionale Sprachen wurden mit diesem Ziel entworfen.

Alle Studenten sind schon von der Mittelschulmathematik mit dem Konzept der Funktion vertraut. Das ist ein Vorteil der Funktionalen Sprachen in der Einführung.

Das imperative Modell

Beginnt man mit einer imperativen Sprache zur Einführung, ist die Situation etwas anders. Alle gängigen imperativen Sprachen sind Abstraktionen der von- Neumann-Architektur (diese Architektur schließt alle verbreiteten Prozessoren ein, von Intel bis zu den Risc-Prozessoren). Die Struktur imperativer Sprachen ist leichter durch deren Hardwarenähe als ausgehend von den zu lösenden Problemen zu rechtfertigen. Deshalb scheint es angebrachter mit der Maschineneben zu beginnen.

In den Studienjahren von 1993 bis 1995 war dies die Basis der Einführung im ersten Semester. In diesem Ansatz würde man aufbauend auf einer vereinfachten Prozessorstruktur (Speicher, Register, Befehlszähler) die Semantik einer repräsentativen Menge von Maschinenin- struktionen durch die entsprechenden Zustandstrans-

formationen definieren. Obwohl dieser Ansatz recht einfach ist, zeigt sich im Unterricht, daß manche der Ideen schwer zu vermitteln sind, Schleifeninvarianten sind ein Beispiel. Vielleicht hat das Prinzip der Induktion, das der Idee der Schleifeninvariante zugrunde liegt, noch nicht ausreichend in den Köpfen Fuß gefaßt. Beginnt man mit einer funktionalen Sprache, muß das Denken in Definitionen und Bewei- sen durch Induktion geübt werden.

Zusammenfassung

Die einführenden Lehrveranstaltungen zur Informatik werden auch an anderen Universitäten mehr oder weniger heftig diskutiert. Aus gegebenem Anlaß, die Studienkommissionenen sind ja dabei neue Studienpläne zu erstellen, wurden einige Gedanken zum laufenden Block der Einführung in die Informatik dargelegt. Persönlich bin ich mehr denn je von der eingeschlagenen Richtung überzeugt. So sehr das spielerische, experimentelle Erlernen des Umgangs mit Geräten und Programmiersprachen zu begrüßen ist und durch entsprechende Räume und Geräte gefördert werden sollte, ist dies letztlich der individuellen Intitative und dem Interesse des Einzelnen überlassen. Dem Unversitätslehrgang obliegt es, die theoretischen Grundlagen und den systematischen Ansatz zu lehren und zu üben.

Das klingt jetzt so, als ob in ersterem das Vergnügen und in letzterem die Mühsal liegt. Dem ist nicht so. Quälend ist, wenn man mit einem halbeverstandenem oder nur halb verstehbaren System solange herumprobieren muß bis man letztlich eine Lösung zu haben glaubt, von der man nicht weiß ob sie hält. Dagegen ist ein befriedigendes Erfolgserlebnis zu gewinnen, wenn aus wohlverstandenen und verstehbaren Elementen, die erstaunlich einfache und daher elegante Lösung gebaut ist und das letzte Argument der Korrektheit mit sattem Klang einschnappt.

Prof. Peter Lucas,
Ordinariat für
Softwaretechnologie

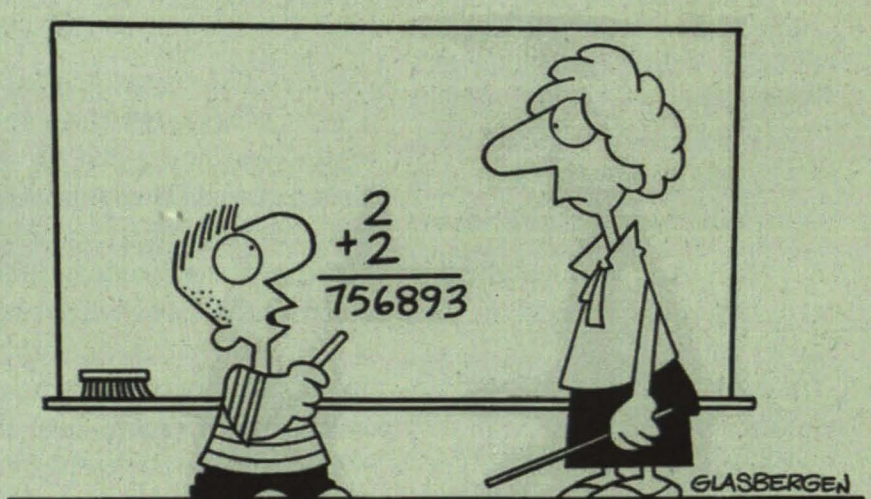
Prof. Peter Lucas

1953 - 1959 Studium der Nachrichtentechnik, TU Wien.

1958 - 1961 Arbeit am „Mailüfterl", ALGOL

Seit 1961 bei IBM Wien, Yorktown Heights, Almaden Research Centre.

Seit WS 1992 /93 an der TU Graz - Ordinarius für Softwaretechnologie.



"In an increasingly complex world, sometimes old questions require new answers."