



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 1

1. Aufgabe (6 Punkte)

Als angehende Informatiklehrerin¹ ist es wichtig, die Wissenschaft Informatik beschreiben zu können. Beantworten Sie die folgenden Fragen, indem Sie u.a. auch auf Ihre Erfahrungen des Informatikstudiums zurückgreifen. Bitte geben Sie Quellen an, wenn Sie sich auf solche beziehen².

- a) Was ist Informatik?
 - eigene Definition
 - Definition für die Hand der Schülerin (Sekundarstufe II – Anfangsunterricht)
- b) Grenzen Sie die Wissenschaft Informatik von anderen Disziplinen ab und begründen Sie ggf. die Notwendigkeit einer Abgrenzung.
- c) Geben Sie zentrale Gegenstände der Informatik an und begründen Sie daran den allgemein bildenden Anspruch der Schulinformatik.
- d) Geben Sie konkrete Beispiele für verschiedene Ebenen an, auf denen Informatik andere Disziplinen unterstützt.

2. Aufgabe (4 Punkte)

In den Empfehlung der Gesellschaft für Informatik (GI) „Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen (September 2000)³ werden die Leitlinien

- Interaktion mit Informatiksystemen
- Wirkprinzipien von Informatiksystemen
- Informatische Modellierung
- Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft

genannt. Führen Sie für jeden der vier Punkte mindestens zwei charakteristische Beispiele auf und begründen Sie den Bezug zur Leitlinie.

Abgabe: 2.5.2003

Besprechung: 8.5.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹ In den Übungsaufgaben wird für personenbezogene Bezeichnungen das sog. „generische Femininum“ verwendet. Die männliche Form ist immer mitgemeint.

² Für die Quellenangaben verwenden Sie bitte den deutschen Zitierstandard DIN 1505, Teil 2 (siehe <http://www.fh-hamburg.de/pers/Lorenzen/bibtex/index.html>)

³ URL: http://www.gi-ev.de/informatik/publikationen/gesamtkonzept_26_9_2000.pdf

„Didaktik der Informatik I“ - Übung 1

1a) Eigene Definition: Informatik ist die Wissenschaft der Informationsverarbeitung durch Informatiksysteme auf der Basis algorithmisch automatisiert geschaffener und veränderter Datenstrukturen.

Definition für Schülerinnen: Informatik ist die Wissenschaft der Informationsverarbeitung durch zumeist elektronische Systeme, die in Form von Befehlsfolgen, Algorithmen, und bestimmten Datenstrukturen automatisiert arbeiten.

b) Die Informatik hebt sich von anderen Wissenschaften durch ihre Eigenschaft als Konglomerat aus vielen verschiedenen Disziplinen ab. So gehen sowohl mathematische, als auch elektrotechnische und physikalische, inhaltlich eng einzugrenzende, Fragestellungen bei der Lösung informatischer Probleme Hand in Hand, so dass die Informatik nicht als Teildisziplin eines einzelnen Wissenschaftsbereich anzusehen ist und ihrem Anspruch als eigenständige Wissenschaft gerecht wird.

c) Informatiksysteme dienen der Lösung von realen, oft auch sozialen Problemstellungen. Aus der Motivation heraus die Produktivität des sozialen und wirtschaftlichen Miteinanders zu optimieren, werden informatische Modelle konstruiert, um Daten einer Bearbeitung zu unterziehen und für die realen Problemstellungen brauchbar zu rekontextualisieren. Die Nutzung von Informatiksystemen durchzieht heutzutage fast alle gesellschaftlichen, wirtschaftlichen und wissenschaftlichen Bereiche. Deren Administration und Programmierung ist somit ein allgemein-bildender Aspekt, der weit über eine ledigliche Betrachtung einer wissenschaftlichen Disziplin hinausgeht.

d) Zum einen können Informatiksysteme Vorgänge unterstützen, die es auch ohne sie gäbe, nur in einem langsameren Prozess von statten gehen würden, z.B. in der Mathematik, zum anderen können sie neue Forschungs- bzw. Arbeitsbereiche überhaupt erst möglich machen, bzw. existierende ersetzen, z.B. bei der Simulation von Atom-Explosionen.

2 Wie schon bereits unter 1c) erwähnt, ist unser gesellschaftliches und berufliches Leben durchzogen mit der Benutzung von, bzw. der Interaktion mit Informatiksystemen. Schülerinnen und Schüler können durch das frühzeitige Erlernen von Interaktion mit Informatiksystemen besser auf ihre berufliche und gesellschaftliche Zukunft vorbereitet werden, z.B. muss selbst ein Klempnerlehrling aller Wahrscheinlichkeit nach lernen, die Buchhaltung der eigenen Arbeit am „PC“ zu erledigen. Der Erwerb von Kenntnissen dieser Art sollte auch in den Informatikunterricht gehören.

Kenntnis über die Wirkprinzipien von Informatiksystemen kann die Vertrautheit und Sicherheit in der Benutzung von Informatiksystemen erleichtern und verbessern, z.B. durch Kenntnis grundlegender Rechnerstrukturen, in Form von Erläuterung verschiedener Begrifflichkeiten wie CPU, Speicher, Eingabe- und Ausgabeinstanzen, etc. Ausserdem wird an dem Punkt, an dem Soft- oder Hardware näher beleuchtet und in seiner Wirkungsweise analysiert wird das unter 1b) beschriebene Disziplinen-Konglomerat deutlich und trägt so zur Allgemeinbildung bei.

Informatische Modellierung wird vor allem die Programmierung im Sinne vom Erlernen und Anwenden algorithmischer Basis-Strukturen aufbauend auf Programmiersprachen wie z.B. Python, beinhalten. Die Anwendung von Programmierkenntnissen lässt sich ihrer Natur nach wohl vor allem in projekt-orientierter Arbeit wiederfinden.

Aufgabe 1

a) Meine Definition des Begriffs der Informatik:

Mein Bild der Informatik ist geprägt durch den Schulunterricht, durch ein Semester Informatikstudiums an der Universität Dortmund und durch wenige visuelle- und schriftliche- Quellen aus Fernsehen und Zeitschriften. Für mich ist die Informatik eine Wissenschaft, welche Geschehnisse der realen Welt modelliert und sie so versucht greifbar und verstehbarer zu machen. Bei einer solchen Modellierung werden Informationen der realen Welt verwaltet und mit Ihnen ein automatisierter, logischer Prozeß in Gang gesetzt (die Algorithmen), welcher am Modell stattfindet und zu Lösungen führt, welche die Realität entweder verständlicher machen oder ein Problem der realen Welt lösen. Daher läßt sich für mich der Begriff Informatik auch ableiten aus den beiden Kernworten 'Information' und 'Automatik'. Diese Modellierung und das automatisierte Rechnen findet im Computer statt, der die Plattform für die Modellierung liefert. Eine genaue, wissenschaftliche Definition gestaltet sich schwierig: Bis heute ist es den Informatikerinnen nicht gelungen, eine eindeutige Definition zu finden.

b) Definition für Schülerinnen:

Für die Schule ist es wichtig, zwischen einem Informatiker und einem Programmierer zu unterscheiden: Häufig stellen sich die Schülerinnen die Informatik anders vor, nämlich vorzugsweise als eine Art Programmierkurs. Das Programmieren nur ein Teilgebiet der Informatik und mehr als Werkzeug denn als elementares Mittel dieser Wissenschaft dient wird den Schülerinnen viel zu selten schon in der Schule klar. Zur Einführung in die Informatik würde ich die Definition des Schülerdudens bevorzugen, mit mündlichen Hinweisen auf die oben genannten Mißverständnisse, die häufig auftreten:

Informatik: Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Digitalrechnern.

Aufgabe 2

Die Informatik wurde hervorgebracht durch die Mathematik, die Elektrotechnik, die Physik und durch die Wirtschaftswissenschaften. Schnell versuchte man durch klare Definition der Informatik diese Wissenschaft von ihren Ursprungswissenschaften abzugrenzen. Der Sinn bestand darin, die Informatik als eigenes Forschungsgebiet im Staat zu etablieren. Laut VOLLMAR ist "[das informatische Vorgehen...] neben theoretischem und experimentellem Vorgehen die dritte Säule der wissenschaftlichen Arbeitsweise." Somit ist die Informatik etwas anderes als ihre Ursprungswissenschaften. Sie ist praktischer als die Mathematik und theoretischer als die Elektrotechnik. Sie bildet Lösungen für Probleme ihrer Ursprungswissenschaften.

Aufgabe 3

Zentraler Gegenstand der Informatik sind die Algorithmen und Datenstrukturen. Ein Algorithmus ist ein mit formalen Mitteln beschreibbares Verfahren zur Lösung von Problemen. In der Schule können somit Verfahren erlernt werden, um physikalische, mathematische, biologische oder andere Probleme zu modellieren und sie so begreifbarer zu machen und natürlich zu lösen. In der Regel sollte der Informatikunterricht in der Schule den Computer als Hilfsmittel begreifbar machen. Es sollte dabei beispielsweise verdeutlicht werden, wie Informatiker Probleme modellieren und dann versuchen zu lösen. Dabei wird der Computer als Werkzeug kennengelernt. Es soll den Schülerinnen klar gemacht werden (gerade in der Mittelstufe), daß ein Computer zu mehr fähig ist als zum Briefe schreiben oder zum Computer spielen. Die Schul informatik dient dabei, den Computer in unserem täglichen Leben vertrauter zu machen. Denn Computer begegnen uns heute überall im Leben.

Aufgabe 4

Die Informatik unterstützte schon in ihren Anfängen die Mathematik bei der schnellen Berechnung komplizierter Formeln. Heute wird sie auch benutzt, um komplexere Probleme zu modellieren und zu lösen. In den Bereichen der Ingenieurwissenschaften dient die Informatik häufig der Modellierung. Durch Modellierung wird die Ingenieurwissenschaft greifbarer und praktischer. Gleiches gilt für die Physik, die Biologie und die Chemie: Wesentliche Fortschritte werden erreicht durch Simulation und Visualisierung. Komplexe Vorgänge werden Verstehbarer und bestehende Prozesse können schneller und besser optimiert werden. Ein konkretes Beispiel läßt sich zum Beispiel in der Autoindustrie finden: Die neuen Autos entstehen dabei am Computer, wo sich am Modell Dinge berechnen lassen wie die Windschnittigkeit des Vehikels. Veränderungen am Fahrzeug können zu Konsequenzen führen, die einem der Computer virtuell vorführt.

Aufgabe 2:

- Interaktion mit Informatiksystemen

Ein Beispiel für interaktives Handeln mit Informatiksystemen im Informatikunterricht ist den SchülerInnen den Umgang mit dem Windows-Explorer nahe zu bringen. Er ist ein Beispiel dafür, wie es den Schülern später möglich ist "die Information, die uns mittlerweile zur Verfügung steht, bewältigen zu können". Die SchülerInnen sollen sich mit Hilfe des Windows-Explorers "einen Vorrat an Grundstrategien und -methoden [an], um Information [] zu strukturieren, zu bearbeiten, aufzubewahren und wieder zu verwenden []". Es wird an einem kleinen Umgebungsraum gelernt um später in "globalen Informationsräumen zu navigieren". Ein zweites Beispiel ist das Internet. Nachdem am lokalen Umgebungsraum anhand des Windows-Explorers Strategien zur Bewältigung von Information angelernt worden sind, kann man die SchülerInnen nun an den globalen Informationsraum Internet heranführen. Hier steht besonders die Beschaffung und Strukturierung der Information im Vordergrund. Letztendlich können die SchülerInnen somit eine eigene Bewertung über die "menschengerechte Gestaltung von Informatiksystemen" abgeben.

-Wirkprinzipien von Informatiksystemen

Um die Wirkprinzipien des globalen Informatiksystems Internet an einem lokalen Umgebungsraum darzustellen, wähle ich als erstes Beispiel das Intranet, welches sich jede Schule einrichten sollte. Anhand der dort gegebenen kleinen Strukturen sollen die SchülerInnen verstehen "nach welchen Funktionsprinzipien [die] Systemkomponenten [des Intranets] zusammenwirken und wie diese sich in größere Systemzusammenhänge [des Internets] einordnen lassen." Dies hilft das Internet zu entmystifizieren. Anhand des Intranets lernen die SchülerInnen grundlegende, prinzipielle Konzepte um später den Aufbau des komplexeren Basissystems Internet zu verstehen. Als zweites Beispiel, die Wirkprinzipien von Informatiksystemen den SchülerInnen beizubringen, wähle ich das Informatiksystem Suchmaschine. Anhand eines einfachen Suchalgorithmus sollen die SchülerInnen die "grundlegenden Ideen und Konzepte" einer Suchmaschine kennen lernen, obwohl deren Wirkungsweise natürlich viel komplexer ist. Mit Hilfe des einfachen Algorithmus, der von Menschen erstellt worden ist, "erfahren [sie] die individuelle Stärkung des Menschen durch die Automatisierung geistiger Tätigkeiten." Ein letztes Beispiel ist, den SchülerInnen anhand der weniger komplexen Strukturen des Taschenrechners die Wirkungsweise des Dualsystems eines Computers näher zu bringen.

-Informatische Modellierung

Modellierung ist die Abgrenzung eines Teils der Realität und die Ausarbeitung deren wichtiger Eigenschaften und der Strukturen.

Beispiele für informatische Modelle sind häufig in der Industrie zu finden. So wird beispielsweise bei der Fertigung von Sportwagen dieser im Computer modelliert. Bestimmte Ausschnitte der Realität wirken dann im Modell virtuell auf das Modell des Sportwagens. So wird zum Beispiel die Luftströmung und der Luftwiderstand genau berechnet.

In der Schule wird als Einführung in die Modellierung oft ein einfacher Automat betrachtet, so zum Beispiel ein Kaffee- / Tee- Automat. Es wird modelliert, welche Eingaben bei einem solchen Automaten möglich und nötig sein sollen, und wie entsprechende Abläufe abgewickelt werden. So kann zum Beispiel per Münzeinwurf entschieden werden, ob jemand Tee oder Kaffee haben möchte, oder per Druck einer Taste, entweder nach oder vor dem Münzeinwurf. Den Schülern stehen hierbei kreative Möglichkeiten zu, welche sich an einem äußerst praktischen Modell verwirklichen lassen.

Ein weiteres, wichtiges Beispiel ist die Modellierung einer Kundenkartei, also einer Datenbank. Der Umgang mit Datenbanken ist gerade in heutiger Zeit bei einfachsten Bürotätigkeiten von größter Bedeutung. Die Schülerinnen lernen dabei nicht nur den Umgang mit Datenbanken, sondern lernen im Zusammenhang mit der Modellierung auch, welche Aufgabe und Möglichkeiten dahinter stecken. Der tägliche Umgang mit Datenbanken wird durch das Verstehen solcher erleichtert. Im Rahmen des Schulunterrichts läßt sich so zum Beispiel eine Kundenkartei in ein virtuelles Geschäft eingliedern, oder eine Kartei einer Arztpraxis erstellen.

Somit lernen die Schüler durch die Modellierung das maschinelle Verarbeiten von Information. Sie lernen, wie sinnvoll der Computer unser tägliches Leben verbessert und wie universell einsetzbar er in der Planung der Wirklichkeit ist.

-Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft

1) Kommunikation Bei der Lösung komplexer Probleme in der Informatik erfahren die Schüler, dass Teamwork an aller erster Stelle steht, wenn es darum geht, Probleme zu bewältigen. Die mehrperspektivische Sicht der Informatik setzt Informationsaustausch und Kooperation voraus. So wird die Kommunikation durch die technischen und medialen Mittel von Informationssystemen unterstützt bzw. teilweise erst ermöglicht.

2) Kreatives Denken und Motivation Durch Informationssysteme als Medium und Werkzeug lernen Schüler kreative Gestaltungsmöglichkeiten kennen mit eigener, selbstverantwortlicher Tätigkeit. Durch systematisches Problemlösen wird beispielsweise das schöpferische Denken gefördert, so dass eigene Modelle entwickelt werden können. So werden das generelle Finden von Lösungsansätzen und der Transfer auf ähnliche Probleme und Inhalte gefördert.

DDI Übung Blatt 1

1. Aufgabe:

a) Was ist Informatik?

- Eigene Definition: Informatik ist die Wissenschaft die sich mit der elektronischen Verarbeitung von Information beschäftigt. Dabei werden Optimierungen in den Bereichen: Datenstruktur, Algorithmus, Rechnerarchitektur, Software- Architektur und Mensch- Maschine Interaktion angestrebt.
- Definition für die Hand von SchülerInnen: In der Informatik werden Methoden und Systeme entwickelt und verbessert mit denen Information gespeichert, verändert und abgerufen werden können. Dabei gibt es verschiedene Zielsetzungen:
 - Technische Entwicklung von Informatiksystemen
 - Strukturelle Entwicklung der Information und Abläufe
 - Anpassung an den Nutzer

Weiter werden auch die Folgen die aus der Nutzung von Informatiksystemen entstehen betrachtet.

b) Abgrenzung der Informatik zu anderen Wissenschaften?

Wie im geschichtlichen Kontext der Informatik begründet, ist eine Abgrenzung von den Wissenschaften Mathematik und Elektrotechnik für ihre Eigenständigkeit notwendig. Einzelne Teilgebiete der Informatik z.B. Informatik und Gesellschaft könnten auch eine Abgrenzung zu anderen Wissenschaften z.B. der Sozialwissenschaft notwendig machen.

Abgrenzung zur Mathematik: Einige grundlegende Ideen und Werkzeuge (Zahlsysteme, Boolesche Algebra, Algorithmen, ..) der Informatik entstammen der Mathematik, dennoch kann die Informatik als eigenständige Wissenschaft und nicht als Spezialgebiet der Mathematik verstanden werden. Die Begründung dafür ist in den Verschiedenen Arbeitsweisen zu sehen. Während in der Mathematik vor allem Logische Konstrukte entwickelt und eingesetzt werden, die nicht notwendig von technischen Hilfsmitteln oder existenten Phänomenen abhängig sind, werden die mathematischen Modelle und Methoden in der Informatik als Hilfsmittel benutzt und weiter entwickelt. Ziel ist der Aufbau eines Systems aus Theorie und technischer Umsetzung und dessen Weiterentwicklung. Hierbei hat in der Informatik die Optimierung und Anpassung der Systeme an praktische Probleme eine hohe Priorität.

Mathematik wird in der Informatik nicht mehr um ihrer selbst Willen betrieben!

Abgrenzung zur Elektrotechnik: Die Informatik bedient sich technischer Methoden und Aufbauweisen um die für sie signifikanten Informatiksysteme konstruieren zu können. Dabei ist sie nicht als Teil der Elektrotechnik zu sehen, da sie eine neue Konstruktion aus Technik und Theorien der Informatik mit eigenen Zielen entwickelt. In der Informatik ist nicht die Technik der Zentrale Aspekt, sondern ein Werkzeug um ihre Ideen damit zu umzusetzen.

c) Zentrale Gegenstände der Informatik

- Information: Abhängig von den verschiedenen Definition des Begriffs Information existieren verschiedene Ansprüche die als allgemein Bildend bezeichnet werden können.
 - Shannonsche Informationstheorie: Liefert eine „mathematische Theorie zur Bestimmung des „Informationsgehalt einer Nachricht“¹.

¹ Vorlesungsskript zur Veranstaltung „Didaktik der Informatik – Teil1“, Seite 9, Stand: 26.03.03

- Nach Christiane Floyd: „Information ist
 - Personal, um Kognition allgemein und insbesondere die Interpretation von Daten durch Menschen zu kennzeichnen,
 - organisationsbezogen, um die roll von Information bei Aktion und Entscheidungsfindung zu zeigen,
 - medial, um Information als eigenständiges, speicherbares und weitergebbares Gut zu betrachten.²

Betrachtet man diese Erläuterungen des Begriffs Information, so ergibt sich der allgemein bildende Anspruch der Schulinformatik aus der Beschäftigung mit Information. Die SchülerInnen lernen hier Dinge die in unserer Gesellschaft gebraucht werden um in Schule, Beruf und Leben klarkommen zu können.

- Algorithmus: Das Erlernen von Strategien mit denen komplexe Probleme in eine handhabbare Abfolge von Handlungen zerlegt werden können, ist ein mächtiges Hilfsmittel im Alltag. Es ermöglicht die Entwirrung oft zu gewaltig erscheinender Aufgaben, die so als einfache Teilaufgaben leicht bearbeitet werden können.
- Datenstruktur: Das Verständnis von Datenstrukturen ist im heute stark von der Informationstechnik abhängigen Arbeitswelt fast unerlässlich. Hier stellt es eine Grundlage für eine große Freiheit in der Berufswahl dar.

d) Beispiele für verschiedene Ebenen, auf denen Informatik andere Disziplinen unterstützt

- In der Chemie bzw. Medizin, wenn aufgrund von Datenbanken und Computersimulationen die Wirkungen von Medikamenten erprobt werden kann. (Software)
- In der Physik, wenn langwierige Vergleiche und Rechnungen durch immer schnellere Informatiksysteme bearbeitet werden können. (Rechnerarchitektur, Datenstruktur, Algorithmen)

2. Aufgabe:

– Interaktion mit Informatiksystemen:

- Erstellen einer Datenbank z.B. über Musik, in der Titel, Interpret, Dauer und Stil angegeben werden. Anschließendes Einbetten der Datenbank in eine Software welche die Suche aufgrund von Einzelaspekten ermöglicht. Nutzen der Suchfunktion im Rahmen eines Klassenprojekts z.B. Organisation eine Feier mit Musik. Betrachten der Vor- und Nachteile innerhalb einer Datenbank ohne weitere Software und mit unterstützender Software zu suchen.
Dieses Beispiel steht im Bezug zu der Leitlinie, da mit dem Erstellen der Datenbank Information *strukturiert, bearbeitet, interpretiert, dargestellt* und *aufbewahrt* wird, mit dem Einbetten in eine Software diese Information zusätzlich *präsentiert* und *wiederverwertet* wird, mit dem Suchen Information *beschafft* und *bearbeitet* wird und schließlich im Vergleich *bewertet* wird.
- Erstellen einer mehrseitigen Webseite z.B. über das Thema „Unsere Klasse“, nachdem zuvor im Internet z.B. über Datenschutz recherchiert wird. Vergleichen der eigenen Seiten mit denen anderer.
Dieses Beispiel steht im Bezug zu der Leitlinie, da hier Aspekte wie Information beschaffen (Datenschutzgesetze), strukturieren (Information über die Klasse), bearbeiten, aufbewahren (speichern), darstellen (Webseite), interpretieren

² Vorlesungsskript zur Veranstaltung „Didaktik der Informatik – Teil1“, Seite 9, Stand: 26.03.03

(Umsetzen der Datenschutzgesetze), bewerten (Vergleich) und präsentieren (Veröffentlichung im Internet) Gegenstand sind.

- Wirkprinzipien von Informatiksystemen:

- Konstruktion eines einfachen Dualen Rechners, bestehend aus mehreren Volladdierern und Flipflops. Dabei kann die Umwandlung Dezimaler Zahlen in Binäre als Beispiel der Kodierung betrachtet werden. Anschließendes schreiben von Befehlsfolgen die von diesem Rechner ausgeführt werden könnten. Dieses Beispiel steht im Bezug zu der Leitlinie, da hier die Grundlagen des Aufbaus von Informatiksystemen visualisiert werden können.
- Erarbeiten des Begriffs Algorithmus und seiner Bedeutung in der automatisierten Lösung gleichartiger Probleme. Zu diesem Zweck kann aus der Frage nach Lösungswegen für z.B. das Suchen einer Zahl innerhalb einer sortierten Liste aus zunächst intuitiven Lösungen eine systematische, sich auf eindeutige Anweisungen beschränkende Lösung in Form eines Algorithmus herausgearbeitet werden. Dieses Beispiel steht im Bezug zu der Leitlinie, da hier sowohl das Prinzip und die Bedeutung eines Algorithmus geklärt werden, als auch Verfahren zur Findung eines Algorithmus erarbeitet werden.

- Informatische Modellierung:

- Erstellen eines Modells für einen Computer. Je nach Ausrichtung kann man eine Zerlegung in Ausgab, Speicher, Verarbeitung und Eingabe oder einen vereinfachten Schaltplan erhalten. Hier wird aus einer Vielzahl von ähnlichen Gegenständen eine Beschreibung abstrahiert, die nur noch für den Zweck grundlegende Dinge enthält.
- Erstellen eines Modells für den Prozess: Finden des billigste Mineralwasser in einem Geschäft und kaufen des selben. Dieser Prozess kann als Finden der kleinsten Zahl innerhalb einer Zahlenfolge modelliert werde. Dieses Beispiel steht im Bezug zu der Leitlinie, da hier durch abstrahieren aus einem speziellen Problem ein Modell entwickelt wird mit dem mehrere gleichartige Probleme dargestellt und im nächsten Schritt im Informatiksystem umgesetzt werden können.

- Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft:

- Diskutieren der Auswirkungen, welche die ungewollte Veröffentlichung privater Briefe haben kann, um an diesem Beispiel den Zweck der Datenschutzgesetze herauszuarbeiten. Anschließende Überlegungen in wie weit der unberechtigte Zugriff auf E-Mails diesem Problem gleich kommt und welchen Zweck in diesem Zusammenhang Verschlüsselung hat. Dieses Beispiel ist Leitlinien gerecht, da hier die Folgen, Chancen und Risiken bei der Nutzung eines Informatiksystems betrachtet und kennen gelernt werden.
- Erarbeiten der Veränderungen des Sozialverhaltens und der Kommunikation bei Menschen, seit dem Beginn der technisch Unterstützten Kommunikation, insbesondere aufgrund von Chat und E-Mail im Vergleich zum direkten Gespräch und zum Handgeschriebenen Brief. Dieses Beispiel steht im Bezug zu den Leitlinien, da hier die Folgen, Chancen und Risiken für den Menschen aufgrund von Informatiksystemen betrachtet werden.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 2

1. Aufgabe (2 Punkte)

Eine der Aufgaben im Zusammenhang mit der Unterrichtsvorbereitung – neben der Auswahl und der Sequenzierung von Lerngegenständen – besteht in der Formulierung der Ziele des Lernprozesses. Lernziele werden häufig in Anlehnung an B. Blooms „Taxonomy of Educational Objectives“¹ nach kognitiven, affektiven und psychomotorischen Lernzielen unterschieden. Sie finden im Folgenden einige Lernziele, die im Zusammenhang mit Unterrichtsentwürfen dazu dienen, die Zielperspektiven einer konkreten Unterrichtsstunde darzustellen:

Die Schülerinnen und Schüler

- überprüfen durch die Transformation eines Datenmodells (gegeben als Entity-Relationship-Diagramm) in ein relationales Datenbankschema die Vollständigkeit des Datenmodells,
- übertragen die Alltagssituation einer Musiktitelverwaltung in ein angemessenes Datenmodell,
- erkennen, dass ein nicht normalisiertes Relationenschema zu inkonsistenten Datenbankeinträgen führen kann,
- empfinden Freude bei der durch die Implementierung realisierte erweiterte Funktionalität ihres Informationssystems,
- erweitern ihre Handlungsmöglichkeit durch einen Export ihrer Datensätze in verschiedene Formate.

(a) Bitte ordnen Sie die Ziele den Kategorien kognitiv, affektiv und psychomotorisch zu.

(b) Nehmen Sie eine Hierarchisierung vor und ordnen Sie die o.g. Ziele den verschiedenen Stufen innerhalb dieser Kategorien zu.

2. Aufgabe (4 Punkte)

Sie finden in der Anlage zu dieser Übung² eine Liste von Verben, die „unter der Hand“ kursiert und eine Möglichkeit der Zuordnung zu den Lernzieltaxonomien darstellt.

Geben Sie aus jedem der insgesamt 16 Unterpunkte je ein Beispiellernziel aus dem Bereich Informationssysteme an und begründen Sie jeweils kurz Ihre Wahl.

3. Aufgabe (4 Punkte)

Gegenwärtig werden mit dem Anliegen, schulisches Lernen und individuelle Persönlichkeitsentwicklung stärker in Beziehung zu setzen, Kompetenzmodelle aufgestellt. Kompetenzen bezeichnen das individuelle Vermögen, durch das das persönliche, berufliche und gesellschaftliche Leben verantwortlich und persönlich befriedigend geführt und mitgestaltet werden kann.

Grenzen Sie die Begriffe „Kompetenzen“ und „Lernziele“ voneinander ab.

Abgabe: 9.5.2003

Besprechung: 15.5.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹Bloom, Benjamin S.: Taxonomy of Educational Objectives. The classification of Educational Goals. New York (David McKay) 1956

²URL: http://inpu.d.cs.uni-dortmund.de/lehre/sose_2003/veranstaltungen4/ddi1/uebungen/anlageblatt2.pdf

Wissen	Verständnis	Anwendung	Analyse	Synthese	Bewertung
angeben	begrenzen	abstellen	ableiten	abfassen	alternieren
antworten	begründen	abstimmen	abschätzen	auffinden	äußern
anzeichnen	charakterisieren	abschließen	abteilen	aufstellen	auswählen
anzeigen	einfügen	ändern	abtrennen	ausarbeiten	auswerten
aufsagen	eingliedern	anknüpfen	analysieren	definieren	bemessen
aufschreiben	einreihen	anpassen	aufdecken	entwerfen	bewerten
aufzählen	einsetzen	anrechnen	auflösen	entwickeln	beurteilen
aufzeichnen	erklären	anwenden	aufspalten	erläutern	deuten
ausführen	erweitern	berechnen	aufspüren	festsetzen	differenzieren
beantworten	fortsetzen	bestimmen	darlegen	gestalten	folgern
benennen	klären	demonstrieren	einengen	lösen	interpretieren
beschreiben	klarlegen	dimensionieren	einkreisen	niederlegen	nuancieren
bezeichnen	klarstellen	durchführen	einordnen	planen	optimieren
darstellen	kürzen	erstellen	einteilen	schließen	prüfen
eintragen	ordnen	ermitteln	erkennen	terminieren	qualifizieren
katalogisieren	präzisieren	formulieren	feststellen	verfassen	reduzieren
kennzeichnen	rezitieren	kalkulieren	gegenüberstellen	verwerfen	Schlüsse ziehen
lesen	schildern	konstruieren	gliedern	zurückführen auf	urteilen
markieren	skizzieren	korrigieren	identifizieren	zusammenfassen	vereinigen
messen	übersetzen	modifizieren	isolieren	zusammenfügen	werten
multiplizieren	übertragen	nutzen	klassifizieren	zusammensetzen	widerlegen
nennen	umstellen	operieren	kontrollieren	zusammenstellen	
niederschreiben	unterscheiden	quantifizieren	lokalisieren	zusammenziehen	
potenzieren	umreißen	realisieren	nachweisen		
rechnen	variieren	sichern	zerlegen		
registrieren	verallgemeinern	steuern	zuordnen		
sammeln	verändern	umgehen	zurechnen		
schreiben	verarbeiten	umgestalten			
skizzieren	verbinden	umschreiben			
summieren	vergleichen	umsetzen			
teilen	wiedergeben	umwandeln			
trennen		verwandeln			
vermehrten		vollbringen			
vermindern		vorgehen			
vervollständigen		zitieren			
wiederholen					
zählen					
zeichnen					
zeigen					

Taxonomie der Lernziele (Bloom 1976, Krathwohl/Bloom/Masia 1978)

Kognitive Lernziele - Was man weiß

Der kognitive Bereich umfasst das Erinnern, die Erkenntnis von Wissen und die Entwicklung intellektueller Fähigkeiten und Fertigkeiten.

- Wissen (im Sinne von Kennen):
 - Reproduzieren des Gelernten
 - Kennen von Verallgemeinerungen, Einzelheiten, Methoden
 - Prozessen, Mustern, Strukturen, Definitionen
- Verstehen:
 - einfachste Form des Begreifens
 - Erfassen vorgegebener Informationen und Benutzen in geringem Umfang
 - Reorganisieren des Gelernten (Inhalt mit eigenen Worten wiedergeben können)
- Anwenden:
 - Abstraktion verstandener Begriffe (allgemeine Erkenntnisse)
 - Übertragung auf neue Probleme (konkrete Situationen)
- Analysieren:
 - Zerlegen von komplexen Zusammenhängen in ihre Bausteine
 - Identifizieren und Analysieren von Beziehungen zwischen ihnen
 - logisches Schließen und Verknüpfen von Fakten und Hypothesen
 - Bestandteil des Problemlösens
- Synthetisieren:
 - kreatives Zusammenfügen bekannter Elemente zu Neuem
 - Entdecken bisher nicht bekannter Gesetzmäßigkeiten
 - Bestandteil des Problemlösens
- Bewerten:
 - Voraussetzung für eigene Entscheidungen
 - Auswerten und Bewerten von Lösungen, Methoden, Ideen zu einem bestimmten Zweck, wobei der Bewertungsmaßstab vorgegeben oder von Schülern selbst festgelegt wird

Affektive Lernziele - Was man will

Der affektive Bereich umfasst die Interessen, Einstellungen und Wertungen.

- Aufmerksamwerden, Aufnehmen, Beachten: Sensibilisieren für bestimmte Phänomene oder Reize, wobei man zwischen der bloßen „Zur-Kennntnisnahme“, der Aufnahmebereitschaft und der gerichteten, gegen Störungen unempfindliche Aufmerksamkeit unterscheiden kann.
- Reagieren: Bereitschaft des Lernenden, seine Aufmerksamkeit aktiv (durch „Mittun“) auf etwas zu lenken.
- Werten:
 - Zuordnen von Werten an und Einschätzung von Reizen, Phänomenen oder Objekten durch den Lernenden
 - Gewinnen von Haltungen und Einstellungen
- Entwickeln von Werte-Strukturen:
 - Erkennen des Konfliktes zwischen mehreren Werten
 - Herstellen von Beziehungen zwischen Werten
 - Überprüfen auf Konsistenz (Beständigkeit)
 - Strukturieren der Werte (z.B. in einer Hierarchie)
- Werte-Verinnerlichung:
 - Aufnehmen von Überzeugungen und Ideen in die eigene Weltanschauung (Philosophie)
 - festes Verankern von Werte-Strukturen durch den Lernenden, dessen Verhalten dadurch weitgehend bestimmt wird
 - Weiterentwickeln der eigenen Weltanschauung

Psychomotorische Lernziele - Was man kann

- Imitation: Nachahmen einer beobachteten Handlung
- Manipulation: Ausführen bestimmter Handlungen nach Anweisung
- Präzision: Verbessern der Genauigkeit von Handlungen, die nun losgelöst vom Vorbild selbst kontrolliert werden
- Strukturierung: Gliedern einer Handlung in Einzelhandlungen und koordiniertes Ausführen
- Naturalisierung: weitgehendes Verlagern des Bewegungsablaufs in das Unterbewusstsein (die Handlung ist in Fleisch und Blut übergegangen)

Kompetenzbegriff der Kultusministerkonferenz

Die Definitionen verschiedener Kompetenzfelder wurden der Quelle (KMK, 2000, S. 9) entnommen:

„Die aufgeführten Ziele sind auf die Entwicklung von **Handlungskompetenz** gerichtet. Diese wird hier verstanden als die Bereitschaft und Fähigkeit des Einzelnen, sich in beruflichen, gesellschaftlichen und privaten Situationen sachgerecht durchdacht sowie individuell und sozial verantwortlich zu verhalten. Handlungskompetenz entfaltet sich in den Dimensionen von Fachkompetenz, Personalkompetenz und Sozialkompetenz.

Fachkompetenz bezeichnet die Bereitschaft und Fähigkeit, auf der Grundlage fachlichen Wissens und Könnens Aufgaben und Probleme zielorientiert, sachgerecht, methodengeleitet und selbständig zu lösen und das Ergebnis zu beurteilen.

Personalkompetenz bezeichnet die Bereitschaft und Fähigkeit, als individuelle Persönlichkeit die Entwicklungschancen, Anforderungen und Einschränkungen in Familie, Beruf und öffentlichem Leben zu klären, zu durchdenken und zu beurteilen, eigene Begabungen zu entfalten sowie Lebenspläne zu fassen und fortzuentwickeln. Sie umfasst personale Eigenschaften wie Selbständigkeit, Kritikfähigkeit, Selbstvertrauen, Zuverlässigkeit, Verantwortungs- und Pflichtbewusstsein. Zu ihr gehören insbesondere auch die Entwicklung durchdachter Wertvorstellungen und die selbstbestimmte Bindung an Werte.

Sozialkompetenz bezeichnet die Bereitschaft und Fähigkeit, soziale Beziehungen zu leben und zu gestalten, Zuwendungen und Spannungen zu erfassen, zu verstehen sowie sich mit anderen rational und verantwortungsbewusst auseinanderzusetzen und zu verständigen. Hierzu gehört insbesondere auch die Entwicklung sozialer Verantwortung und Solidarität.

Eine ausgewogene Fach-, Personal-, Sozialkompetenz ist die Voraussetzung für **Methoden- und Lernkompetenz.**“

Übungsblatt 2

1. Aufgabe

a)

Lernziele - Unterrichtsentwürfe:

- a. Überprüfen der Vollständigkeit eines Datenbankmodells durch die Transformation des Datenmodells (gegeben als Entity-Relationship-Diagramm) in ein relationales Datenbankschema

Kognitive Lernziele:	-Wissen	-Kennen von Einzelheiten, Methoden
	-Verstehen	-Kennen von Mustern, Strukturen, Definition
	-Anwenden	-Erfassen vorgegebener Informationen und Benutzen in geringem Umfang
	-Analysieren	-Abstraktion verstandener Begriffe (allgemeine Erkenntnisse)
	-Synthetisieren	-Zerlegen von komplexen Zusammenhängen in ihre Bausteine (in dem Entity-Relationship-Diagramm)
	-Bewerten	-Identifizieren und Analysieren von Beziehungen zwischen ihnen

Affektive Ziele -Aufmerksamwerden

Psychomotorische Lernziele

- Präzision: Verbessern der Genauigkeit von Handlungen, die nun losgelöst vom Vorbild selbst kontrolliert werden
- Strukturierung: Gliedern einer Handlung in Einzelhandlungen und koordiniertes Ausführen
- Naturalisierung: Verinnerlichung

- b. Übertragen die Alltagssituation einer Musiktitelverwaltung in ein angemessenes Datenmodell

Kognitive Lernziele	-Wissen	-Kennen von Methoden
	-Verstehen	-Erfassen vorgegebener Informationen und Benutzen in geringem Umfang
	-Anwenden	-Übertragen auf konkrete Situationen
	-Analysieren	-Zerlegen in die Bausteine
	-Synthetisieren	-Identifizieren von Beziehungen zwischen ihnen

Affektive Lernziele -Reagieren -Bereitschaft des Lernenden, seine Aufmerksamkeit aktiv (durch Mittun) auf etwas zu lenken

Psychomotorische Ziele -Präzision
-Strukturierung
-Naturalisierung

- c. Erkennen, dass ein nicht normalisiertes Relationenschema zu inkonsistenten Datenbankeinträgen führen kann

Kognitive Lernziele	-Wissen	-Kennen von Verallgemeinerungen, Einzelheiten, Methoden
		-Mustern, Strukturen, Definitionen

- Analysieren -Zerlegen von komplexen Zusammenhängen in ihre Bausteine
- Analysieren von Beziehungen zw. Ihnen
- logisches Verknüpfen von Fakten
- Bewerten
- Affektive Lernziele -Aufmerksamwerden, Beachten: Sensibilisieren für bestimmte Phänomene
- Werteverinnerlichung
- Psychomotorische Lernziele
- Präzision -Verbessern der Genauigkeit von Handlungen

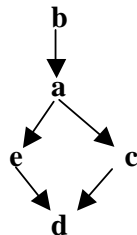
d. Empfinden Freude bei der durch die Implementierung realisierte erweiterte Funktionalität ihres Informationssystems

- Affektive Lernziele -Aufmerksamwerden
- Reagieren
- Werten
- Werte-Verinnerlichung

e. Erweitern ihre Handlungsmöglichkeit durch einen Export ihrer Datensätze in verschiedene Formate

- Kognitive Lernziele -Wissen -Kennen von Verallgemeinerungen, Einzelheiten, Methoden
- Anwenden -Übertragung auf neue Probleme
- Synthetisieren-kreatives Zusammenfügen bekannter Elemente zu Neuem
- Affektive Lernziele -Aufmerksamwerden
- Werten -Gewinnen von Haltungen und Einstellungen
- Psychomotorische Lernziele
- Präzision
- Naturalisierung Verinnerlichung von Handlungen

Hierarchisierung



2. Aufgabe

Kognitiven Lernziele

1. Wissen zeigen

Einfügen eines Elementes in ein Array

2. Verständnis einsetzen

Grafische Darstellungen – „Geometrie entdecken mit dem Computer“

3. Anwendung nutzen

Suchmaschine benutzen können

4. Analysieren
PC-Hardware und Betriebssysteme
5. Synthese - zusammenfügen
Erstellung einer Datenbank für eine Schulverwaltung
6. Bewertung – bemessen, auswählen
Darstellung eines Datenbankmodells als Hierarchisches Datenbankmodell, Vernetztes Datenbankmodell und relationales Datenbankmodell. Anschließend: Vergleich und Bewertung des flexibelsten Modells
Affektive Lernziele
7. Aufmerksamwerden, Aufnehmen, Beachten
Simulation eines Ökosystems – Ein einfaches Modell eines begrenzten Lebensraumes, an dem biologisches Gleichgewicht und dessen Störungen durch menschliche Eingriffe verdeutlicht werden können
8. Reagieren
Kreative Gestaltung mit dem Computer in verschiedenen Sachbereichen (z.B. Musik, Kunst, etc.)
9. Werten
Zweckgebundene, kritische und selbständige Auswahl von Software
10. Entwickeln von Werte-Strukturen
Beurteilung der Auswirkungen der Informations- und Kommunikationstechniken unter Berücksichtigung des Jugend- und Datenschutzes
11. Werte – Verinnerlichung
Sensibilisierung für Fragen der sozial verträglichen Technikgestaltung
Psychomotorische Ziele
12. Imitation
Einfache Texte am PC verfassen und gestalten
13. Manipulation
Gestalten und Verarbeiten von anspruchsvolleren Texten und Grafiken am PC
14. Präzision
Erweiterung der Grundkenntnisse in der PC-Bedienung durch den sicheren Umgang mit Daten sowie Text- und Grafikprogrammen
15. Strukturierung
Erstellung eines Programms, das eine unsortierte Schülerdatei nach Schülernamen sortiert.
16. Naturalisierung
Erstellung eines Programms zur statischen Auswertung der Schülerdatei: Durchschnittsnote: allgemein, pro Klasse, etc.

3. Aufgabe

Die Lernziele beziehen sich direkt auf den Unterrichtsstoff. Diese sind „jetzt, in der Gegenwart“ des schulischen Lebens gegliedert.
Handlungskompetenz visiert die Entfaltung der Persönlichkeit jetzt und in der „Zukunft“, im späteren Berufs- und Sozialleben.
Natürlich gibt es eine Interdependenz zwischen den Lernzielen und der Handlungskompetenz. Die Lernziele sind auf die Entwicklung von Handlungskompetenzen gerichtet und die Grundlagen der Handlungskompetenz sind gerade die kognitiven, affektiven und psychomotorischen Lernziele.
Fachkompetenz heißt nicht Aneignung, Assimilieren vom Wissen, die Schüler müssen im späteren Berufsleben in der Lage sein, das fachbezogene und fachübergreifende Wissen anzuwenden, selbständig Probleme zu lösen.

Aufgabe 1:

- a) -Kognitive Lernziele: Die SchülerInnen
- o übertragen die Alltagssituation einer Musiktitelverwaltung in ein angemessenes Datenmodell,
 - o erkennen, dass ein nicht normalisiertes Relationenschema zu inkonsistenten Datenbankeinträgen führen kann,
 - o erweitern ihre Handlungsmöglichkeit durch einen Export ihrer Datensätze in verschiedene Formate.

-Affektive Lernziele: Die SchülerInnen

- o empfinden Freude bei der durch die Implementierung realisierte erweiterte Funktionalität ihres Informationssystems.

-Psychomotorische Lernziele: Die SchülerInnen

- o überprüfen durch die Transformation eines Datenmodells (gegeben als Entity-Relationship-Diagramm) in ein relationales Datenbankschema die Vollständigkeit des Datenmodells.

b) Das Lernziel, dass die SchülerInnen die Alltagssituation einer Musiktitelverwaltung in ein angemessenes Datenmodell übertragen, entspricht im Allgemeinen dem kognitiven Lernziel "Anwenden". Hier erfolgt eine Übertragung allgemeiner Erkenntnisse auf neue Probleme.

Das Lernziel, dass die SchülerInnen erkennen, dass ein nicht normalisiertes Relationenschema zu inkonsistenten Datenbankeinträgen führen kann, lässt sich dem kognitiven Lernziel "Analysieren" zuordnen. Die SchülerInnen schließen logisch und verknüpfen Fakten und Hypothesen.

Das Lernziel, dass die SchülerInnen ihre Handlungsmöglichkeit durch einen Export ihrer Datensätze in verschiedene Formate erweitern, ist ein Teil des kognitiven Lernziels "Synthetisieren". Durch das Zusammenfügen bekannter Elemente zu Neuen, erweitern sie ihren Horizont mit bislang nicht bekannten Gesetzmäßigkeiten.

Das Lernziel, dass SchülerInnen Freude bei der durch die Implementierung realisierte erweiterte Funktionalität ihres Informationssystems empfinden, entspricht dem affektiven Lernziel "Werten". Sie ordnen ihre Gefühle den im Unterricht besprochenen Themen zu und gewinnen somit eine neue Haltung gegenüber dem Unterricht.

Das Lernziel, dass die SchülerInnen durch die Transformation eines Datenmodells (gegeben als Entity-Relationship-Diagramm) in ein relationales Datenbankschema die Vollständigkeit des Datenmodells überprüfen, lässt sich dem psychomotorischen Lernziel "Präzision" zuordnen. Sie kontrollieren losgelöst vom gegebenen Datenmodell die Vollständigkeit des Datenmodells.

Aufgabe 2:

Kognitive Beispiellernziele aus dem Bereich Informationssysteme:

Wissen (im Sinne von Kennen): Bsp.: Kenntnis von Java - Befehlen:

Als Basis für die Lösung einer Programmieraufgabe ist es wichtig, die entsprechenden Befehle zu kennen. Ohne sie ist die Bewältigung des Problems unmöglich, so dass diese Grundlage an Wissen essentiell notwendig ist. Der Schüler reproduziert das Gelernte und macht sich so Stück für Stück mit den Prinzipien der Programmierung vertraut.

Verstehen: Bsp.: Interpretation von einem Algorithmus:

Hier dient ein einfacher Algorithmus dazu, beispielsweise den Nutzen der Rekursivität zu begreifen oder einfach nur zu verstehen, was der Algorithmus an sich für eine Funktion hat. Hierdurch begreifen die Schüler nicht nur den Aufbau eines kleinen Programms, sondern sehen auch neue Methoden zur Lösung eines Problems.

Anwenden: Bsp.: Tabellenkalkulation

Die Schüler wurden mit den einzelnen Befehlen und der Benutzungsoberfläche eines Programms für Tabellenkalkulation vertraut gemacht. Nun wenden sie diese Grundlage an, um beispielsweise einen Haushaltsplan mit der Summe aller Ausgaben aufzustellen. Somit wird das Wissen durch die Anwendung erweitert und praktische Erfahrung kommt ebenfalls hinzu.

Analysieren: Bsp.: Java - Programm:

Durch das Zerlegen eines Java - Programms, sieht der Schüler, in welcher Weise die einzelnen Befehle miteinander agieren, um hinterher zu dem gewünschten Resultat zu kommen. Somit werden neue Zusammenhänge deutlich und neue Wege zur effektiven Programmierung erlernt.

Synthetisieren: Bsp.: Fortsetzung Java - Programm:

Nach der Analyse kann man nun die neu erworbenen Fähigkeiten einsetzen, um selbstständig eine Programmieraufgabe zu lösen, oder alte Aufgaben zu verfeinern bzw. zu verbessern. Hierdurch werden Probleme in einem neuen Blickwinkel betrachtet und neue Methoden zur Lösung entdeckt.

Bewerten: Bsp.: Effizienz von einem Algorithmus:

Die Schüler können selbst bewerten, ob der Algorithmus verbesserungswürdig ist, oder ob er schon in seinem jetzigen Zustand gut arbeitet. Somit lernen sie die "richtige" Programmierung eines solchen Algorithmus zu schätzen, dass sie effizienter ist und entwickeln eigene Ideen zur Optimierung eines solchen Programms.

Affektive Beispiellernziele aus dem Bereich Informationssysteme:

Aufmerksamwerden, Aufnehmen, Beachten: Bsp.: Interesse an Computern
Schüler nehmen den Reiz der modernen Informationssysteme wahr und nehmen keine negative Haltung diesen gegenüber ein. Gleichzeitig reicht diese Art von "Beachtung schenken" jedoch nicht aus, um sich intensiv mit Computern zu beschäftigen und in die Tiefe der Materie vorzudringen. So glauben z.B. auch viele, sie hätten Ahnung von Informatik, nur weil sie sich etwas mit Computern auskennen.

Reagieren: Bsp.: In der Freizeit programmieren
Durch die freiwillige Aktivität des Lernenden wird gezeigt, dass er Interesse für die in der Schule behandelten Inhalte empfindet und sich gerne mit ihnen auseinandersetzt. Hierdurch wird das selbstständige Handeln des Schülers gefördert.

Werten: Bsp.: Der Informatik einen Platz in seinem Leben einräumen
Zunächst sollte der Wert von der Person angenommen werden, indem er der Informatik Vertrauen schenkt und sie als nützliche Wissenschaft in der Gesellschaft ansieht. Der Lernende identifiziert sich dann mit dem Wert und lernt ihn zu bevorzugen. So versucht er dann später auch die Informatik nach außen hin zu repräsentieren und andere von ihr zu überzeugen.

Entwickeln von Werte - Strukturen: Bsp.: Die Stellung der Informatik in der Gesellschaft rational fundieren können
Als ersten Schritt wird versucht die Eigenschaften der Informatik zu begreifen, wodurch eine gewisse Abstraktion stattfindet. Dann wird die Informatik mit anderen Werten in der Gesellschaft verglichen, so dass Beziehungen zwischen ihr und einzelnen Werten aufgebaut werden. So kann die Informatik einen gewissen Stellenwert in der Hierarchie der Werte in der Gesellschaft erhalten, welche bei jedem Individuum anders ist.

Werte - Verinnerlichung: Bsp.: Achtung vor dem Wert der Informatik
Hierbei ist es wichtig die richtigen Entscheidungen zu treffen und welche Konsequenzen diese für die Gesellschaft haben könnten. Informatik kann nämlich durchaus auch für kriminelle Zwecke von großem Nutzen sein. So muss jeder Mensch die Werte der Informatik achten und ist für seine Handlungen selbst verantwortlich.

Psychomotorische Beispiellernziele aus dem Bereich Informationssysteme:

Imitation: Bsp.: Nachahmen eines Short - Cut - Befehls in Word

In der Mittelstufe wird ja zuerst bekanntermaßen Textverarbeitung gelehrt. Hier kann ein Schüler beispielsweise durch einfaches Beobachten der Aktionen anderer Mitschüler lernen, wie man diese ausführt. Wozu die Befehle gut sind, weiß er zunächst nicht, lernt dieses jedoch dann durch das Resultat, was bei der Ausführung hinauskommt.

Manipulation Bsp.: Die Anleitung zur Installation eines Betriebssystems

Der Lernende führt hier seine Handlungen nicht etwas durch Beobachten aus, sondern allein nach der Instruktion des Buches. Jedoch kann hierdurch die Manipulation bestimmter Geräte gelernt werden, wenn Installationen z.B. nach demselben Schema ablaufen.

Präzision: Bsp.: Programmierstil

Der Lernende verbessert stetig seine eigenen Programmierfertigkeit und kann so nach einer gewissen Zeit unabhängig vom Lehrer selbst programmieren. Desweiteren präzisiert er seine eigenen Kenntnisse immer weiter, indem er einen eigenen Programmierstil entwickelt, ohne jedoch die vorher gelernten Parameter zu verletzen.

Strukturisierung: Bsp.: Entwicklung eines Programms

Wenn man ein Programm schreiben möchte, sollte man nie sofort anfangen los zu programmieren. Erst sollten eine Reihe von Analysen und Vorüberlegungen getroffen werden. Somit findet hier eine strukturierte Herangehensweise an das eigentliche Problem statt.

Naturalisierung: Bsp.: Schreiben mit der Tastatur

Das Schreiben mit der Tastatur wird in solchem Maße zur Routine, dass sie in eine automatische und von sich selbst ablaufende Reaktionsfolge übergeht.

Aufgabe 3

Ziel des Schulunterrichts ist im allgemeinen, den Schülerinnen Kompetenzen anzueignen, welche ihnen bei privaten, komplexen oder kreativen Problemlösungen helfen. Aufgabe der Lehrerin ist es zu erkennen, welche Aufgaben eine Schülerin später besser lösen können soll. Diese Aufgabe wird auf die entsprechende Kompetenzen hin analysiert. Es werden im nächsten Schritt Lernziele formuliert, mit denen versucht wird, die gewünschten Kompetenzen der Schülerin zu fördern.

Die Lernziele dienen also nur als Instrument, um gewisse Kompetenzen bei Personen aufzubauen. Kognitive Lernziele werden meist benutzt, um die Fachkompetenz zu fördern. Affektive Lernziele dienen sowohl der Förderung Personalkompetenz als auch der Sozialkompetenz. Psychomotorische Lernziele hingegen dient meist wieder der Förderung der Fachkompetenz. Jedoch sind die Grenzen, welche Lernziele welche Kompetenzen fördern können, nicht eindeutig sondern fließend.

Folgende Definition halte ich in diesem Zusammenhang für sinnvoll: Quelle: Brandenburger Bildungsministerium: Hinweise zum Unterricht im Modellversuch (1994) (<http://www.evfh-berlin.de/evfh-berlin/html/download/oe/mitarbeiter/hauptkraft-friedhelm/LER.pdf>)

”Kompetenzen beschreiben, welche Eigenschaften oder Fähigkeiten der Lernende schließlich erreichen soll. Insofern stehen Kompetenzen über den Lernzielen, die eher besagen, welche konkreten Ziele Unterricht im einzelnen verfolgen muß, um solche Eigenschaften und Fähigkeiten zu befördern.”

Als Beispiel im Schulunterrichts möchte eine Lehrerin bei ihren Schülerinnen die Fachkompetenz erhöhen: Die Schülerinnen sollen also die Fähigkeit erlernen, mit fachlichem Wissen ein gewisses Problem selbstständig zu lösen, beispielsweise durch den Umgang mit einer gewissen Programmiersprache. Die Lehrerin versucht nun zuerst Wissen zu vermitteln, in der Form, daß die Schülerinnen die Syntax und die Grammatik der neuen Programmiersprache erlernen. Die Schülerinnen erkennen dabei Muster und Strukturen und lernen neue Methoden kennen. Nach der Übermittlung des Wissens geht es um das Verstehen des Wissens. Dabei sollen zum Beispiel kleine Programmfragmente mehr Klarheit verschaffen. Als weiterer Schritt kann die Lehrerin nun ein Problem aufstellen, welches die Schülerinnen mit Hilfe der neu gelernten Programmiersprache lösen sollen. Hierbei geht es darum, daß Erlernte auch anzuwenden und die gelernten Befehle bei anderen Problemen selbst zu implementieren. Wichtige Bestandteile hierbei sind auch das Analysieren und Synthetisieren, welche zur Problemlösung beitragen sollten. Als Abschluß einer solchen Lernzielphase, in welcher kognitive Lernziele benutzt wurden, sollte es den Schülerinnen möglich sein, die neu gelernte Programmiersprache zu bewerten, Vorteile und Nachteile gegenüber anderen Programmiersprachen aufzuzeigen und ihre Nutzbarkeit einzuschätzen. Mithilfe von kognitiven Lernzielen wird in unserem klassischen Beispiel also versucht, die Fachkompetenz im Bereich der Programmierfähigkeit mit jener gewissen Programmiersprache zu erhöhen.

Übung 2 Didaktik der Informatik I

1. Aufgabe:

a / b)

kognitiv	affektiv	psychomotorisch
Wissen: überprüfen durch die Transformation eines Datenmodells (geben als Entity- Relationship-Diagramm) in ein relationales Datenbankschema die Vollständigkeit des Datenmodells	Werten: empfinden Freude bei der durch die Implementierung realisierte erweiterte Funktionalität ihres Informationssystems	Manipulation: erweitern ihre Handlungsmöglichkeit durch einen Export ihrer Datensätze in verschiedene Formate
Anwenden: übertragen die Alltagssituation einer Musiktitelverwaltung in ein angemessenes Datenmodell		
Synthetisieren: erkennen, dass ein nicht normalisiertes Relationenschema zu inkonsistenten Datebankeinträgen führen kann	Entwickeln von Werte-Strukturen: erkennen, dass ein nicht normalisiertes Relationenschema zu inkonsistenten Datebankeinträgen führen kann	

2. Aufgabe:

- **Kognitive Lernziele:**

- **Wissen :** Erlerntes Wissen aus einer Programmiersprache in einer anderen Programmiersprache umzusetzen. Jeder der Programme entwickelt darf nicht nur in einer Programmiersprache arbeiten, sondern muss sein Wissen auch in andere Programmiersprachen umsetzen können.
- **Verstehen :** Die Zusammenhänge von Dateien, Ordnern und Laufwerken. Als Grundlage muss jedem Anwender bekannt sein, wie er mit diesen Begriffen umzugehen hat und wie er Funktionen darauf anwenden kann.
- **Anwenden :** Der Umgang mit Anwenderprogrammen. Es ist nicht nur wichtig die theoretischen Funktionen von Anwenderprogrammen zu kennen sondern auch die Umsetzung auf ein aktuelles, reales Problem.
- **Analysieren :** Die Fehlersuche in einem Programm. Jeder der als Programmierer arbeitet oder auch nur den Fehler in einem Programm suchen will, muss das Programm Stück für Stück analysieren um bestimmte Programmkomponenten von Fehlern ausschließen zu können.
- **Synthetisieren :** Die Entwicklung eines Programms. Um ein neues Programm zu entwickeln müssen nicht immer alle Komponenten des Programms neu geschrieben werden, sondern es können bestimmte Programmkomponenten übernommen bzw. verändert werden.

- **Bewerten** : Jeder soll selber erkennen können, ob eine Programmieraufgabe korrekt gelöst wurde. Es ist wichtig, dass jeder Schüler selber erkennt, ob die Lösung eines Problems ausreichend bzw. den Anforderungen entsprechend ist.
- **Affektive Lernziele**
 - **Aufmerksamwerden, Aufnehmen, Beachten** : Die SchülerInnen dafür sensibilisieren, was eine gute Software ausmacht und das diese nicht notwendig die am meisten genutzte sein muss.
 - **Reagieren** : Beteiligen an Lösungssuche zu einer Problemstellung. Der Schüler/ die Schülerin zeigt Interessen an der Fragestellung.
 - **Werten** : Der Schüler/ die Schülerin macht positive Erfahrungen im Umgang mit einer Programmiersprache und gewinnt dadurch an Aufgeschlossenheit für diese Sprache.
 - **Entwickeln von Werte- Strukturen** : Die Vor- und Nachteile gefundener Lösungen werden gegeneinander abgewogen. Dazu wird eine Hierarchie der einzelnen Aspekte aufgestellt, so dass die für die Situation bestmögliche Lösung ausgewählt werden kann.
 - **Werte- Verinnerlichung** : Wertungen zu einzelnen Softwaresystemen werden übernommen und zu einer eigenen Meinung über diese Softwaresysteme weiterentwickelt.
- **Psychomotorische Lernziele**
 - **Imitation**: Das Einrücken von Quellcode Zeilen und Abschnitten beim Programmieren wird übernommen.
 - **Manipulation** : Den SchülerInnen wird eine Problemzerlegung vorgegeben nach der sie ein Problem strukturieren sollen.
 - **Präzision** : Eigene Ideen zur Entwicklung von Informatiksystemen werden in die vorgegebenen Handlungsmuster eingebettet um diese zu optimieren.
 - **Strukturierung** : Der Prozess vom Problem zum Problem lösenden Programm wird in Einzelschritte, wie Abstraktion des Problems, Modellieren des Problems, Entwerfen einer Lösung,... , Umsetzen der Lösung und Testen der Lösung, zerlegt.
 - **Naturalisierung** : Sichere Kenntnisse der benutzten Programmiersprache, so dass keine Zeit für die Überlegung, wie eine entwickelte Programmstruktur in der entsprechenden Sprache gut umgesetzt werden kann, aufgewendet werden muss.

3. Aufgabe:

Mit dem Begriff Lernziele werden Ziele des Unterrichts bezeichnet. Dabei wird der Gesamtkontext des Unterrichts betrachtet in dem sowohl die Klasse sowie jeder einzelne Schüler, jede einzelne Schülerin Lernziele erreichen soll. Diese Lernziele werden im Vorfeld festgelegt und sind von außen durch Leitlinien motiviert.

Die Lehrperson führt ihren Unterricht Zielorientiert aus, d.h. sie versucht die SchülerInnen an das Lernziel heranzuführen. Dazu werden einzelne Aufgaben und Projekten bearbeitet mit denen jeweils Teile des Gesamtziels erreicht werden sollen.

Die Fertigkeiten und das Wissen das sich jeder Schüler, jede Schülerin dabei aneignen kann ist seine, ihre Kompetenz.

Auch wenn SchülerInnen am gleichen Unterricht teilgenommen haben werden die in diesem Rahmen erlangten Erkenntnisse und Fähigkeiten verschieden sein. Daraus ergibt sich das die Kompetenz und Lernziel nicht äquivalent sein können.

Anders als bei den Lernzielen die innerhalb einer Klasse gleich belegt sind, ist die Kompetenz ein individuelles Ergebnis für jeden Schüler, jede Schülerin.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 3

Wie in der Vorlesung angesprochen, gibt es Dokumentationen von Ergebnissen durchgeführter Projekte. Andererseits werden Handreichungen für die Durchführung projektorientierten Unterrichts angeboten. Selten werden die im Zusammenhang mit der konkreten Durchführung des Projektunterrichts (oder projektorientierten Unterrichts) auftretenden Schwierigkeiten dokumentiert.

1. Aufgabe (5 Punkte)

- (a) Geben Sie eine Liste aller Kriterien an, die erfüllt sein müssen, damit Unterricht als Projektunterricht bezeichnet werden kann.
Ordnen Sie die Kriterien nach ihrer Bedeutung. 2 Punkte
- (b) In den Richtlinien für Informatik in der gymnasialen Sekundarstufe II¹ wird auf Projektunterricht hingewiesen.
Die Zuordnung zur zeitlichen Verortung wurde im Laufe der Zeit verändert.
Geben Sie an, worin diese Veränderung besteht und nennen Sie Konsequenzen für die informatische Bildung. 3 Punkte

2. Aufgabe: (5 Punkte)

- (a) Geben Sie wesentliche Gemeinsamkeiten und Unterschiede zwischen einem Projekt der Softwareentwicklung und einem Projekt in der Schule an. 2 Punkte
- (b) Geben Sie Beispiele an, an denen die „Problemzonen“, aber auch die Chancen des Projektunterrichts deutlich werden. 3 Punkte

Abgabe: 16.5.2003

Besprechung: 22.5.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹Bitte lesen sie im Skript (Kapitel 7), wie sie eine lokale Kopie des Lehrplans beziehen können.

Didaktik der Informatik Übungsblatt 3

Aufgabe 1:

a)

Folgende Kriterien müssen erfüllt sein, damit Unterricht als Projektunterricht bezeichnet werden kann:

- problemhaltige Aufgabe
- Motivation
- Zielorientierung
- Planvolles, selbständiges und selbstbestimmtes Lernen
- Verbindung von schulischem und außerschulischem Tun
- Hingabe und ernstes Engagement
- individuelles und kooperatives Handeln
- Verbindung von Theorie und Praxis
- Ausdauer
- Abbau von Lehrerdominanz
- typische Verlaufstruktur
- fachübergreifender Charakter
- Methodenvielfalt
- Abschluss und Aufgabenbeurteilung
- gesellschaftliche Relevanz

Ich denke, dass die Motivation bei projektorientiertem Unterricht von zentraler Bedeutung ist, sie geht aus mehreren Faktoren hervor: Zum einen muss der Lehrer eine problemhaltige Aufgabe stellen, wodurch auch die Zielorientierung mit ins Spiel kommt welche eine wichtige Voraussetzung für die Motivation ist. Die Motivation wird auch durch die gesellschaftliche Relevanz eines Projektes erzeugt bzw. bestimmt. Hier geht auch der fachübergreifende Charakter eines Projektes ein, der die Verbindung von schulischem und außerschulischem Tun bezweckt, da er die gesellschaftliche Relevanz mit prägt. Außerdem geht die Hingabe und das ernste Engagement aus der Motivation hervor.

Die wichtige Ziele des Projektunterrichts sind hingegen: Planvolles und selbständiges Lernen, die Verbindung von Theorie und Praxis, individuelles und kooperatives Handeln, den die Lösung von Problemen durch Methodenvielfalt und das Lernen von typische Verlaufstrukturen. Der Abschluss und die Aufgabenbeurteilung eines Projektes sollten jedoch nicht überbeurteilt werden, da das wichtige an projektorientiertem Unterricht der Prozess eines Projektes ist und nicht dessen Ergebnis.

Zuletzt ist noch zu erwähnen das der Abbau von Lehrerdominanz mit einem Projektunterricht einher gehen muss, da die Eigenständigkeit der Schüler bei einem Projekt sonst behindert würde.

b)

Die „Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasium/Gesamtschule in Nordrhein-Westfalen – Informatik“ äußern sich wie folgt zur zeitlichen Verortung des Projekt-Unterrichts:

„Da solche Projektveranstaltungen stufenspezifische Ziele verfolgen, sind sie im Hinblick auf die Teilnehmerinnen und Teilnehmer in der Regel auf eine Jahrgangsstufe oder auf die gymnasiale Oberstufe zu beschränken.“¹

Daraus folgt, dass der Projektunterricht nicht als die einzig mögliche und „ultimative“ Unterrichtsform gelten soll, sondern nur einen Teil der Unterrichtswirklichkeit ausmachen soll.

Aufgabe 2:

a)

Ein Projekt in der Schule ist in sofern von einem Projekt in der Softwareentwicklung zu differenzieren, da in der Schule nicht produktorientiert gearbeitet wird. Das Ziel eines Projektes in der Schule sollte nicht primär das konkrete Ergebnis sein sondern der Weg dort hin, bzw. die eigentliche Arbeit an einem Projekt.

Die Kooperation ist in beiden Bereichen von äußerster Wichtigkeit, jedoch sollte in der Schule die in der Softwareentwicklung oft herrschende Delegation von Aufgaben nicht vorkommen, statt dessen sollte eine Aufteilung vorgenommen werden welche ein Prinzip der Partnerschaft beherbergt. Ebenso ist in der Schule das Lernen bei einem Projekt im Vordergrund zu sehen, wohingegen es in der Softwareentwicklung mehr als Mittel zu Zweck gesehen wird.

Gemeinsamkeit sind mehr in der Frage nach der Art der Umsetzung zu finden. Es muss kooperativ gearbeitet werden und es muss selbstbestimmt gelernt werden um die Problemstellung zu lösen. Die Verlaufstrukturen stimmen auch größtenteils überein.

Auch das fächerübergreifende (disziplinübergreifende) Arbeiten ist bei beiden Arten von enormer Bedeutung, genauso wie die genutzte Methodenvielfalt.

Ich denke das trotz der vielen Gemeinsamkeiten die beiden Arten von Projekten klar von einander differenziert werden müssen.

b)

Die idealistischen Vorzüge projekthafter Arbeit, wie Freiheit, Selbstständigkeit und Kooperation unter den Schülern sind zugleich die Schwachstellen. Es ist problematisch die Schüler in ihrer Arbeit zum gewünschten Ziel zu „lenken“, die Kontrolle über den Arbeitsablauf zu bewahren und vor allem auch diesen zu bewerten und die durch Gruppenarbeit erschwerte Benotung vorzunehmen.

¹ Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasium/Gesamtschule in Nordrhein-Westfalen – Informatik S. 41

Blatt 3

Aufgabe 1a:

Bezieht man sich zunächst nur auf Heinz Pütt so sind die Folgenden Kriterien als notwendig für Projektunterricht zu nennen:

- planvolles, selbstständiges und selbstbestimmtes Lernen
- Verbindung von Theorie und Praxis
- fachübergreifender Charakter

Diese Kriterien sollten aber um Ideen aus dem Modell „ Schritte und Merkmale eines Projektes“ von Herbert Gudjons² ergänzt werden:

- Situationsbezug der Aufgabe
- Praxisrelevanz der Aufgabe
- Zielgerichtete Projektplanung
- Selbstorganisation und Selbstverantwortung bei der Entwicklung von Lösungen
- Produktorientierung im Zusammenhang mit dem Erfahren von Grenzen des Projektunterrichts

Ordnet man diese Kriterien nach ihrer Bedeutung so ergibt sich folgende Reihenfolge: (wichtig -> weniger wichtig)

1. planvolles, selbstständiges und selbstbestimmtes Lernen
2. Verbindung von Theorie und Praxis
3. fachübergreifenden Charakter
4. Zielorientierung
5. Motivation
6. problemhaltige Aufgabe
7. Hingabe und ernstes Engagement
8. Verbindung von schulischem und außerschulischem Tun
9. individuelles und kooperatives Handeln
10. Ausdauer
11. Abbau von Lehrerdominanz
12. Abschluss und Aufgabenbeurteilung
13. gesellschaftliche Relevanz
14. Methodenvielfalt
15. typische Verkaufsstruktur

Aufgabe 1b:

Wir haben zwar den aktuellen Lehrplan finden können, konnten aber keine Veränderungen, die sich „im Laufe der Zeit“ ergeben haben, erkennen.

Aufgabe 2a:

Gemeinsamkeiten:

- es kann eine feste Einteilung der Arbeitsaufgaben geben
- bei beiden muss die Fertigstellung des Programms zu einem bestimmten Abgabetermin eingehalten werden
- die Teamarbeit spielt eine wichtige Rolle und wird dadurch gefördert

- der Entwicklungsprozess der Software, also Planung, Einteilung der Aufgaben, zusammensetzen der Teillösungen, erstellen eines Handbuchs, usw. ist in beiden Fällen sehr ähnlich

Unterschiede:

- ein Projekt in der Softwareentwicklung hat finanzielle Folgen
- arbeitet ein Mitarbeiter bei der Softwareentwicklung nicht mit, kann es sein, dass er entlassen wird; in der Schule bekommt der Schüler höchstens "eine" schlechte Note
- in der Schule kann die Lehrerin zur Not noch helfen
- in der Softwareentwicklung gibt es durch den Lohn einen zusätzlichen Anreiz, das Programm fertig zu stellen, in der Schule muss die Motivation, die durch die Lehrerin gegeben wird, ausreichen

<i>Projekt der Softwareentwicklung</i>	<i>Gemeinsamkeiten</i>	<i>Projekt in der Schule</i>
ausschließlich Produktorientiert		Die Entwicklungsschritte sind wichtiger als das Ergebnis
	Zerlegen einer Aufgabe in Teilbereiche	
In unbeschönigter Form	Praxisrelevante Aufgaben	Oft beschönigt
	Aufgaben bezogenes Weiterbilden	
Entwickler und Anwender sind verschiedene Personengruppen		SchülerInnen sind Entwickler und Anwender
Thema ist immer Softwareentwicklung		Projekt in der Schule muss nicht auf Softwareentwicklung beschränkt sein

b)

- Bei der Bearbeitung eine Aufgabe tauchen immer wieder Probleme auf, welche die SchülerInnen zu immer neuen Lösungsentwürfen zwingen. (Dabei wird deutlich das Problemlösen kein linearer prozess ist, die SchülerInnen gewinnen an Sicherheit in der Strucktuierung von Problem und Lösungsentwurf.)
- Aufgaben werden den Fähigkeiten der SchülerInnen entsprechend verändert, so dass der praxisbezug verloren geht.
- Probleme durch einen zeitlichen Rahmen, in dem das Projekt bearbeitet werden kann treten auf. (SchülerInnen sind an einem Punkt angelangt von dem eine Lösung greifbar ist und die Stunde/ das Schuljahr ist zu Ende.)
- Die SchülerInnen haben den Freiraum projektbezogen Inhalte selbstständig zu erarbeiten. Dabei können sowohl Erkenntnisse aus dem Projekt als auch Zusatzwissen zur Projektbearbeitung erworben werden.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 4

Auf der zweiten Seite finden Sie eine Abituraufgabe für einen GK Informatik.
Dieses Übungsblatt (inkl. Anhang) darf keinesfalls weitergegeben werden.
Es ist ausschließlich für die Bearbeitung in der entsprechenden Übung erstellt worden.

Material: aktuelle EPA Mathematik¹ (enthält Aufgabenvorschläge mit Lösungen)

1. Aufgabe (5 Punkte)

Bearbeiten und lösen Sie die Abituraufgabe.

Beschreiben Sie im Aufgabenteil e) den Algorithmus zunächst verbal, bevor Sie ihn in Python² implementieren. Achten Sie darauf, dass keinerlei herstellerspezifische Erweiterungen benutzt werden dürfen, damit die Programme auf allen Klienten, für die die gewählte Sprache zur Verfügung steht (stehen wird!) lauffähig sind.

2. Aufgabe (1 Punkt)

Erstellen Sie einen einleitenden Aufgabentext, der die Schülerinnen in den Kontext einführt.

3. Aufgabe (3 Punkte)

Welche unterrichtlichen Gegenstände müssen zur Bearbeitung der Aufgabenteile (bitte differenziert beantworten) thematisiert worden sein, damit die Schülerinnen die Aufgabe bewältigen können.

4. Aufgabe (1 Punkt)

Nehmen Sie Stellung zu dem Problem, englischsprachige Dokumente als Teil einer Abiturklausur Informatik einzusetzen.

Abgabe: 23.5.2003

Besprechung: 6.6.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹http://www.kmk.org/doc/beschl/epa_mathematik.pdf

²<http://www.python.org/>

Die folgende Aufgabe darf **unter keinen Umständen** weitergegeben werden.

Ausschnitt aus einer Abituraufgabe (Grundkurs Informatik):

In dieser Abituraufgabe geht es um eine einfache Klient-Server-Applikation, die dafür sorgt, dass ein Informatiksystem im lokalen Netz die Zeit zur Verfügung stellt, während alle anderen auf diesen Service zugreifen.

Als Grundlage wird das Originaldokument verwendet (als Material für die Schülerinnen)
<http://www.faqs.org/rfcs/rfc868.html>

Aufgabentext (ohne Einleitung – siehe Übungsblatt 2. Aufgabe):

- a) Wozu werden in vernetzten Umgebungen abgestimmte Zeiten benötigt?
Welche Gründe sind ursächlich für die Zeitdifferenzen in Informatiksystemen verantwortlich?
- b) Welche Lösungsmöglichkeit wird in dem RFC vorgeschlagen?
Stellen Sie die Kommunikation zwischen Server und Klient in geeigneter Weise grafisch dar.
- c) Erläutern Sie die Unterschiede zwischen dem UDP und dem TCP-Protokoll, geben Sie typische Anwendungsfälle mit der entsprechenden Protokollauswahl an und stellen Sie Vor- und Nachteile der jeweiligen Wahl zusammen.
- d) Erweiterung: Beschreiben Sie die Vor- und Nachteile verbindungsloser Protokolle am Beispiel der Internettelefonie.
- e) Implementieren Sie prototypisch einen Server und einen Klienten, der dem RFC 868 genügt. Benutzen Sie dazu einen Port, der von jeder Benutzerin geöffnet werden kann. Geben Sie zunächst einen „verbalen“ Algorithmus an, den Sie anschließend in Python implementieren.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 5

1. Aufgabe: (6 Punkte)

Diese Aufgabe knüpft inhaltlich an Aufgabe 1 des nicht öffentlichen Übungsblatts 4 an. Ihre Aufgabe bestand darin, eine für den Informatikunterricht typische Abituraufgabe zu lösen. Entwickeln Sie nun eine Bewertungsskala für eine Lösungsskizze dieser Aufgabe, aus der hervorgeht, wie viele Punkte für die korrekte Lösung einer Teilaufgabe vergeben werden sollte und eine Zuordnung dieser Punkte zu den Anforderungsniveaus I, II und III. Orientieren Sie sich hierbei an der aktuellen EPA Mathematik¹.

2. Aufgabe (4 Punkte)

Behauptung:

"Die insgesamt schwache Positionierung und das Zurückfallen im Zeitablauf geben genügend Hinweise, dass es in Deutschland an der Neigung der Unternehmen fehlt, in Qualifikation und Weiterbildung zu investieren. Dies gilt auch und gerade für den Bereich der Informatik, wo sich das Wissen besonders schnell entwertet"

[Grupp (ISI) u. a. 2003, S. 35]²

Als Beleg für diese Aussage wird

http://www.gi-ev.de/informatik/presse/presse_021203.shtml
angeführt.

Stimmen Sie mit dieser Behauptung überein?

Begründen Sie ihre Position!

Abgabe: 30.5.2003

Besprechung: 5.6.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹http://www.kmk.org/doc/beschl/epa_mathematik.pdf

²[Grupp (ISI) u. a. 2003] GRUPP (ISI), Hariolf ; LEGLER (NIW), Harald ; GEHRKE (NIW), Birgit ; BREITSCHOPF (IWW), Barbara ; BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG (BMBF) (Hrsg.): Bericht zur technologischen Leistungsfähigkeit Deutschlands – 2002. Februar 2003. ISI – Fraunhofer- Institut für Systemtechnik und Innovationsforschung, Karlsruhe, NIW – Niedersächsisches Institut für Wirtschaftsforschung, Hannover, IWW – Institut für Wirtschaftspolitik und Wirtschaftsforschung der Universität Karlsruhe http://www.niw.de/publikationen/gutachten/2003/02_2003/tlf_2002.html (geprüft 13.5.2003) http://www.technologische-leistungsaehigkeit.de/_htdocs/tlf_58.php (geprüft 13.5.2002)

Aufgabe 1:

Abituraufgabe:

a)

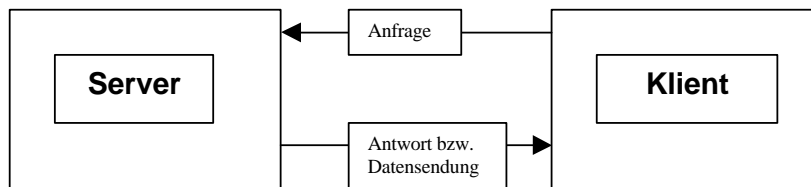
In vernetzten Umgebungen werden häufig aufeinander abgestimmte Zeiten benötigt. Oft kommt es vor, dass nicht alle Informatiksysteme innerhalb einer vernetzten Umgebung eine date- oder time-clock besitzen, so dass sie von zufällig auftretenden menschlichen oder technischen Fehlern abhängig sind.

Um bei solch auftretenden Fehlern das Netz aufrecht zu erhalten bzw. die Fehler zu beseitigen, werden so genannte Time-Server eingesetzt, welche jeden Standort innerhalb des Netzwerks kurz „abtasten“.

b)

Um die oben genannten Zeitdifferenzen zu beseitigen, werden die wie schon in Teil a) beschriebenen Time-Server eingesetzt. Die korrekte Zeit wird von diesem Server an die Klienten gesendet und somit auf jedem Klient korrigiert.

Graphisch: (allgemeine Kommunikation zwischen Server und Klient, nicht spezifisch nach dem TCP oder UDP, genauer siehe Teil c))



c)

Der grundlegende Unterschied zwischen dem TC- und dem UD-Protokoll liegt darin, dass das TC-Protokoll ein Protokoll mit fester Verbindung zwischen Server und Klient ist und das UD-Protokoll ein verbindungsloses Protokoll ist.

Beim TCP besteht eine direkte Verbindung zwischen Server und Klient, d.h. der Server wartet auf Anfragen und der Klient verbindet sich mit dem Server. Der Server sendet die angeforderten Daten direkt zum Klient, dieser empfängt sie und der Verbindungsabbruch erfolgt vom Klient.

Beim UDP hingegen besteht erst gar keine direkte Verbindung zwischen Server und Klient. Der Server wartet auf und empfängt Anfragen, die in Form von leeren Datenpaketen vom Klient kommen. Daraufhin sendet der Server ein Datenpaket an den Klient, das die angeforderten Daten (hier: Zeit als 32 Bit Binärzahl) enthält.

UDP findet man z.B. bei folgenden Diensten: Network Time Protocol, NFS Server, DNS. Das Hauptproblem beim UDP besteht in der Sicherheit. Da keine direkte Verbindung besteht, bekommt der Server keine Auskunft darüber ob, die gesendeten Daten richtig und vollständig empfangen wurden, es könnten Daten verloren gehen oder von dritten bearbeitet worden sein. Der größte Vorteil des UDP ist, dass es mit viel weniger Aufwand und einfacher zu realisieren ist als das TCP.

Das TCP ist beispielsweise bei folgenden Diensten vorzufinden: World Wide Web, Extended File Name Server, FTP.

Die Sicherheit des TCP ist natürlich durch die direkte Verbindung viel höher als die des UDP, jedoch ist der Aufwand, um es realisieren zu können um einiges höher.

d)

Bei der Internettelephonie besteht die Gefahr, dass Daten verloren gehen. Da keine direkte Verbindung besteht, werden immer Datenpakete gesendet und es wird nicht sichergestellt, ob diese vollständig übermittelt werden. So könnten beispielsweise Worte eines Satzes vom „Netz verschluckt werden“ und beim Empfänger nicht ankommen. Zudem ist nicht gewährleistet, dass andere Personen das Gespräch nicht abhören können.

Der Vorteil der Internettelephonie liegt im Preis. Es ist viel billiger über ein verbindungsloses Protokoll Daten zu versenden (hier: zu Telefonieren), als über das Festnetz.

e)

Bei diesem Aufgabenteil mussten wir leider feststellen, dass uns die Grundlagen fehlen, um überhaupt einen Algorithmus dieser Art zu erstellen zu können.

Aufgabe 2:

Der Einleitende Text einer Abituraufgabe sollte möglichst knapp ausfallen, damit keine Zeit mit dem Einleiten einer Aufgabe verschwendet wird. Es sollte trotzdem ein direkter Fingerzeig auf den vorher behandelten Stoff hergestellt werden, daß die Schülerinnen also sofort einordnen können, um welchen Stoff es sich handelt, welcher in dieser Aufgabe abgefragt wird bzw. wann und wie dieser Stoff im Unterricht durchgenommen wurde. Außerdem sollte allein mit dem einleitenden Text schon klar sein, welches Themengebiet in der folgende Aufgabe abgefragt wird. Falls vorhanden sollte auf evtl. Anlagen hingewiesen werden.

Unser Vorschlag für einen einleitenden Text (hier wird davon ausgegangen, daß keine Anlage zur Aufgabenlösung beiliegt):

"Ein wichtiges Thema der Informatik ist die Kommunikation zwischen Klient und Server. In diesem Zusammenhang treten, wie im Unterricht behandelt (*), Überlegungen und Begriffe auf wie das im RFC868 vorgestellte Time Protokoll, die Synchronisation von Klient und Server und die UDP und TCP Protokolle. Lösen sie zu auf Basis dieser Daten die folgenden Aufgaben."

Uns fehlen ferner die Kenntnisse über Schul-Software, um die Thematik im Unterricht anschaulich darzustellen und zu veranschaulichen. Mit dem Wissen, wie etwas veranschaulicht wurde, könnte man in der Aufgabenstellung darauf hinweisen.

Schön wäre solch ein Zusatz, wenn man Beispielsweise mit einer Simulation namens XYZ eine Klient-Server Applikation dargestellt hätte. Ein solcher Zusatz könnte an der Stelle (*) wie folgt eingefügt werden:

(*) "...und am Beispiel der XYZ Simulation dargestellt..."

Aufgabe 3:

Zur Bearbeitung der Aufgabenteile müssen folgende Gegenstände im Unterricht thematisiert worden sein:

- Zu a): Um diese Frage zu beantworten muß zunächst einmal ein bestimmtes Grundlagenwissen vorhanden sein. Die Schülerinnen müssen generell wissen, was ein Netzwerk ist und wie eine sog. Netzwerkumgebung aussieht, da es sich hier um eine auf den Stoff aufbauende Frage handelt. Desweiteren müssen im Unterricht verschiedene Server-Typen, wie z.B. hier Time-, Clock- und Date – Server, behandelt und ihre Funktionsweise geklärt worden sein. Dadurch kann die Schülerin dann selbstständig folgern, wozu diese in einer vernetzten Umgebung benötigt werden. Damit der zweite Teil der Frage beantwortet werden kann, muß die Schülerin sich über Gefahren und Risiken im Netzwerkaufbau im Klaren sein. Dadurch kann sie selbst einschätzen zu welchen Problemen es zwischen den verschiedenen Informatiksystemen kommen kann.
- Zu b): Hierzu muß bekannt sein, was ein RFC – Server ist und wie er funktioniert. Dieser muß im Kontext zu dem Netzwerkaufbau besprochen worden sein, damit die Schülerinnen über die Lösungsmöglichkeit Bescheid wissen. Wenn der Begriff ausreichend im Unterricht definiert und geklärt wurde, dürfte es somit auch kein Problem sein, das Ganze grafisch darzustellen.
- Zu c): Diese Frage bezieht sich auf die verschiedenen Protokoll – Typen (TCP und UDP) in Netzwerken. Zur Beantwortung muß bekannt sein, was die Abkürzungen bedeuten und was die Unterschiede im Detail sind. Anwendungsgebiete müssen unserer Meinung nach nur beispielhaft im Unterricht erwähnt worden sein, da die Schülerin durch einzelne Beispiele dann selbst auf andere Fälle folgern kann. Vor- Und Nachteile der einzelnen Anwendungsfälle der entsprechenden Protokollauswahl sind so speziell von den Beispielen der Schülerin abhängig. Dieses „Pro und Contra“ muß jedoch auch nicht vorher intensiv besprochen worden sein, da die Schülerinnen durch die Kenntnis der Unterschiede zwischen den einzelnen Protokolltypen selbst eine Argumentationsweise entwickeln können.
- Zu d): Zur Beantwortung dieser Frage müssen lediglich die Inhalte bekannt sein, die bereits im Aufgabenteil zuvor gefordert worden sind. Zusätzlich präzisiert werden sollte das Thema Internettelefonie, so daß sich jede Schülerin etwas unter diesem Begriff vorstellen kann. Wir sehen diese Aufgabe eher im Licht der selbstständigen Arbeit, d.h. daß die Schülerin die Gedankengänge aus der zuvor behandelten Aufgabe weiterspinn und sie so auf ein anderes Problem überträgt.
- Zu e): Zur Implementierung muß selbstverständlich die Sprache Python bekannt sein. Diese muß vorher in diversen anderen Situationen im Unterricht benutzt und erklärt worden sein. Zusätzlich muß die Schülerin zuvor verschiedene Algorithmen gesehen und ihre Arbeits- bzw. Funktionsweise sollte vom Lehrkörper erklärt worden sein. Natürlich ist hier auch wiederum das Verständnis des hier zentralen Themas (Netzwerkaufbau und Funktionsweise mit verschiedenen Servertypen) notwendig.

Aufgabe 4:

Hauptproblem bei dem Einsatz von Englischsprachigen Texten in einem Fach wie Informatik sind natürlich die unterschiedlichen Sprachkenntnisse der Schülerinnen. Gerade bei anspruchsvollerer, englischer Fachliteratur ist oft ein größeres Vokabular notwendig, um den gesamten Inhalt zu verstehen und ihn dann auch anwenden zu können. Schülerinnen, welche starke Defizite im Schulfach Englisch aufweisen, sind schon bei einem "normalen" Englischen Text gegenüber anderen benachteiligt, und können daher entsprechende Aufgaben in der Informatik schlechter bzw. langsamer lösen als andere. Defizite im Fach Englisch sollten sich jedoch eigentlich nicht in einem anderen Fach wie Informatik auswirken.

Andererseits ist die meiste und beste Fachliteratur –gerade im Bereich Informatik- nur auf Englisch zu haben, und somit ist der Umgang mit englischen Vorlagen für einen guten Informatiker wichtig. Außerdem kann man mit dem Argument des fächerübergreifenden Unterrichts englische Originaltexte legitimieren.

Nach kurzer Diskussion ist sich unsere Gruppe schnell einig geworden, daß wir dem Einsatz englischer Texte in einer Abiturklausur im Fach Informatik eher skeptisch gegenüberstehen. Zwar ist es durchaus legitim, manchmal einen englischen Text im Unterricht oder evtl. auch in einer "normalen" Klausur einzureichen, jedoch sollten in einer (abschließenden) Abiturklausur die Schwerpunkte der Wissenschaft thematisiert werden. Und da ein gutes englisches Sprachverständnis zwar hilfreich und wünschenswert ist, jedoch nichts mit der eigentlichen Informatik zu tun hat, darf es in keiner Weise den Ausgang einer Abiturklausur beeinflussen. In anderen Klausuren ist es hingegen durchaus legitim, wenn man (nicht zu oft) eine leichte, gut verständliche englische Beilage, welche als Hilfestellung dienen sollte, zur Klausur hinzufügt. Ein fundamentales Grundwissen im Bereich Englisch, mit welchem sich einfachere Texte ohne Probleme verstehen lassen sollten, kann vorausgesetzt werden.

Übungsblatt 5

Aufgabe 1:

Im Folgenden wird eine Bewertungsskala für unsere Lösung der Abituraufgabe vorgeschellt. Um dies möglichst nahe an unserer Lösung klar zu machen, steht in Klammern immer die Lösung des entsprechenden Aufgabenteils, jeweils im Anschluss die kommentierte Einteilung des Aufgabenteils in die drei verschiedenen Anforderungsbereiche. Am Ende erfolgen ein Fazit und die genaue Punkteverteilung auf die einzelnen Teilaufgaben.

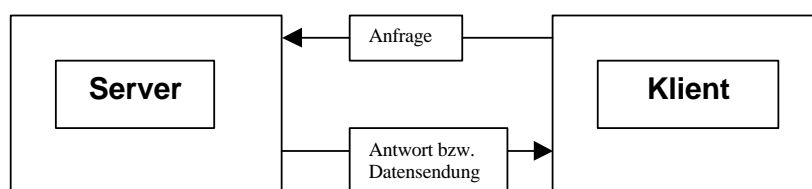
[a) In vernetzten Umgebungen werden häufig aufeinander abgestimmte Zeiten benötigt. Oft kommt es vor, dass nicht alle Informatiksysteme innerhalb einer vernetzten Umgebung eine date- oder time-clock besitzen, so dass sie von zufällig auftretenden menschlichen oder technischen Fehlern abhängig sind.

Um bei solch auftretenden Fehlern das Netz aufrecht zu erhalten bzw. die Fehler zu beseitigen, werden so genannte Time-Server eingesetzt, welche jeden Standort innerhalb des Netzwerks kurz „abtasten“.]

Aufgabenteil **a)** lässt sich dem **Anforderungsbereich 1** zuordnen. Die SchülerInnen haben einen Text als Grundlage. Die Lösung des Aufgabenteils steht fast wörtlich in diesem Text. Die SchülerInnen haben im Unterricht gelernt, Text zu bearbeiten und aus diesen nützliche Informationen zu ziehen. Somit erfolgt hier die „Verwendung gelernter und geübter Arbeitstechniken und Verfahrensweisen in einem begrenzten Gebiet und in einem wiederholendem Zusammenhang.“¹

[b) Um die oben genannten Zeitdifferenzen zu beseitigen, werden die wie schon in Teil a) beschriebenen Time-Server eingesetzt. Die korrekte Zeit wird von diesem Server an die Klienten gesendet und somit auf jedem Klient korrigiert.

Graphisch: (allgemeine Kommunikation zwischen Server und Klient, nicht spezifisch nach dem TCP oder UDP, genauer siehe Teil c))



]

Aufgabenteil **b)** besteht aus zwei Teilfragen, wobei die erste Frage aus dem **Anforderungsbereich 1**, die zweite dem **Anforderungsbereich 2** stammt. Die Antwort auf die Frage nach den Lösungsmöglichkeiten in dem RFC lassen sich

¹ http://www.kmk.org/doc/beschl/epa_mathematik.pdf

ebenfalls aus dem Text herausarbeiten. Im zweiten Teil müssen die SchülerInnen allerdings selbstständig das Gelernte auf eine vergleichbare neue Situation übertragen², sie müssen selbstständig die Kommunikation zwischen Server und Klienten graphisch darstellen. Dies gehört zu den folgenden Teilgebieten des Anforderungsbereichs 2: „Veranschaulichen und Beschreiben von Zusammenhängen bei bekannten Sachverhalten mit Hilfe von Bildern, Texten und Symbolen.“¹ „Übersetzen [...]eines Funktionsterms in eine Skizze“¹

Den SchülerInnen ist im Text die Kommunikation zwischen Server und Klient als Dialog gegeben, sie müssen diese graphisch skizzieren.

[c) Der grundlegende Unterschied zwischen dem TC- und dem UD-Protokoll liegt darin, dass das TC-Protokoll ein Protokoll mit fester Verbindung zwischen Server und Klient ist und das UD-Protokoll ein verbindungsloses Protokoll ist.

Beim TCP besteht eine direkte Verbindung zwischen Server und Klient, d.h. der Server wartet auf Anfragen und der Klient verbindet sich mit dem Server. Der Server sendet die angeforderten Daten direkt zum Klient, dieser empfängt sie und der Verbindungsabbruch erfolgt vom Klient.

Beim UDP hingegen besteht erst gar keine direkte Verbindung zwischen Server und Klient. Der Server wartet auf und empfängt Anfragen, die in Form von leeren Datenpaketen vom Klient kommen. Daraufhin sendet der Server ein Datenpaket an den Klient, das die angeforderten Daten (hier: Zeit als 32 Bit Binärzahl) enthält.

UDP findet man z.B. bei folgenden Diensten: Network Time Protocol, NFS Server, DNS. Das Hauptproblem beim UDP besteht in der Sicherheit. Da keine direkte Verbindung besteht, bekommt der Server keine Auskunft darüber ob, die gesendeten Daten richtig und vollständig empfangen wurden, es könnten Daten verloren gehen oder von dritten bearbeitet worden sein. Der größte Vorteil des UDP ist, dass es mit viel weniger Aufwand und einfacher zu realisieren ist als das TCP.

Das TCP ist beispielsweise bei folgenden Diensten vorzufinden: World Wide Web, Extended File Name Server, FTP.

Die Sicherheit des TCP ist natürlich durch die direkte Verbindung viel höher als die des UDP, jedoch ist der Aufwand, um es realisieren zu können um einiges höher.]

Der Aufgabenteil **c)** entspricht den **Anforderungsbereichen 1 und 2**. Zum einen müssen die SchülerInnen nur die im Text beschriebenen Unterschiede zwischen den beiden Protokollen wiedergeben (Anforderungsbereich 1). Auf der anderen Seite müssen sie allerdings dann auch selbstständig das im Unterricht und durch den Text Gelernte übertragen, indem sie spezielle Anwendungsfälle der beiden Protokolle darlegen und hieran Vor- und Nachteile bestimmen. Sie „übertragen das Gelernte auf vergleichbare neue [Fälle].“¹(Anforderungsbereich 2).

[d) Bei der Internettelephonie besteht die Gefahr, dass Daten verloren gehen. Da keine direkte Verbindung besteht, werden immer Datenpakete gesendet und es wird nicht sichergestellt, ob diese vollständig übermittelt werden. So könnten beispielsweise Worte eines Satzes vom „Netz verschluckt werden“ und beim Empfänger nicht ankommen. Zudem ist nicht gewährleistet, dass andere Personen das Gespräch nicht abhören können. Der Vorteil der Internettelephonie liegt im Preis. Es ist viel billiger über ein verbindungsloses Protokoll Daten zu versenden (hier: zu Telefonieren), als über das Festnetz.]

² vgl. http://www.kmk.org/doc/beschl/epa_mathematik.pdf

Die **Anforderungsbereiche 2 und 3** werden durch diesen Aufgabenteil **d)** abgedeckt. Durch Unterricht und den vorliegenden Text haben die SchülerInnen verbindungslose Protokolle kennen gelernt; hier müssen sie einerseits das Gelernte anwenden, indem sie die Vor- und Nachteile dieser Protokolle darstellen, andererseits aber auch völlig selbstständig und kreativ sich Vor- und Nachteile der Internettelephonie überlegen und diese in Verbindung mit verbindungslosen Protokollen bringen. Somit erfolgt also einerseits ein „selbstständiges Auswählen[...]und Darstellen bekannter Sachverhalte unter vorgegebenen Gesichtspunkten in einem durch Übung bekannten Zusammenhang“¹(Anforderungsbereich 2), andererseits aber auch ein „bewusstes und selbstständiges Auswählen [...] geeigneter gelernter Methoden und Verfahren in neuartigen Situationen“¹(Anforderungsbereich 3).

```
[ e) # File:timeserver.py
import socket
import struct, time
# user-accessible port
PORT = 8037
# reference time
TIME1970 = 2208988800L
# establish server
service = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
service.bind(("", PORT))
service.listen(1)
print "listening on port", PORT
while 1:
    # serve forever
    channel, info = service.accept()
    print "connection from", info
    t = int(time.time()) + TIME1970
    t = struct.pack("!I", t)
    channel.send(t) # send timestamp
    channel.close() # disconnect
```

```
# File:client.py
import socket
import struct, time
# local timeserver "localhost" (you can also insert local ip-address)
# please try remote host "haspe.homeip.net" as well
HOST = "localhost"
PORT = 8037
# reference time (in seconds since 1900-01-01 00:00:00)
TIME1970 = 2208988800L # 1970-01-01 00:00:00
# connect to server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
# read 4 bytes, and convert to time value
t = s.recv(4)
t = struct.unpack("!I", t)[0]
```



```

t = int(t - TIME1970)
s.close()
# print results
print "server time is", time.ctime(t)
print "local clock is", int(time.time()) - t, "seconds off" ]

```

Diese Lösung des Aufgabenteils **e)** gehört dem **Anforderungsbereich 3** an. Hier liegt eine komplexe Problemstellung vor: Die SchülerInnen sollen selbstständig einen Server und einen Klienten erst verbal, dann in der Programmiersprache Python implementieren. Die Erstellung des „verbalen“ Algorithmus lässt sich zwischen den Anforderungsbereichen 2 und 3 anordnen, die Implementierung in Python dann aber vollständig dem Anforderungsbereich 3. Dieser Aufgabenteil umfasst „bewusstes und selbstständiges Auswählen und Anpassen geeigneter gelernter Methoden und Verfahren in neuartigen Situationen.“¹

Fazit:

Man kann eindeutig erkennen, dass „das Schwergewicht der zu erbringenden Prüfungsleistungen im Anforderungsbereich II liegt und die Anforderungsbereiche I und III berücksichtigt werden, und zwar Anforderungsbereich I in höherem Maße als Anforderungsbereich III.“¹

Nun haben wir unsere eigene Punkteverteilung für diese Aufgabe erstellt: Davon ausgehend, dass wir insgesamt 30 Punkte für diese Aufgabe verteilen, sieht die entsprechende Verteilung auf die einzelnen Aufgabenteile folgendermaßen aus: Um nahe an den Vorlagen der EPA-Mathematik zu bleiben, teilen wir die Punkte zunächst in die drei Anforderungsbereiche ein:
 Anforderungsbereich 1 – 10 Punkte,
 Anforderungsbereich 2 – 13 Punkte,
 Anforderungsbereich 3 – 7 Punkte.

Nun die Aufteilung der Punkte auf die einzelnen Aufgabenteile:

Aufg.-Teil	Teilgebiet	Anf.-Bereich	Punkte
a)	- Gründe für abgestimmte Zeiten	1	2,5
	- Gründe für Zeitdifferenzen	1	2,5
b)	- Lösungsmöglichkeit	1	3
	- Graphische Darstellung	2	3
c)	- Unterschiede	1	2
	- Anwendungsfälle mit Vor- und Nachteilen	2	4
d)	- Allgemein: Vor- und Nachteile verbindungsloser Protokolle	2	3
	- Am Beispiel: Internettelefonie	3	3
e)	- „verbaler“ Algorithmus	2	3
	- Implementierung in Python	3	4
Macht insgesamt		1	10
		2	13
		3	7

Aufgabe 2:

Für einen Studenten ist es nicht ganz leicht, hierzu Stellung zu nehmen. Wir (unsere Gruppe) haben noch nie fest in einem Betrieb gearbeitet und haben daher wenig Einblicke in die Berufsförderung innerhalb oder auch außerhalb der Betriebe. Wir können daher nur teilweise aus "eigener Erfahrung" sprechen und sind vielmehr auf Medienberichte und Erfahrung anderer angewiesen. Einschätzen lässt sich die Weiterbildung in Deutschland trotzdem leicht.

Wir stimmen der Behauptung zu. Es ist richtig, dass in Zeiten der Globalisierung vor allem die Konkurrenz der Firmen eine große Rolle spielt. Damit eine Firma konkurrenzfähig bleibt, müssen die Mitarbeiter natürlich immer "auf dem neusten Stand" sein. Und in den meisten Betrieben, gerade in denen, in welchen sich Strukturen schnell ändern, müssen somit regelmäßig Fortbildungen angeboten werden, damit sie konkurrenzfähig bleiben.

Weiterbildung in Deutschland findet statt. Jedoch ist sogar uns, die wir noch nie eine "Weiterbildung" genossen haben, aufgefallen, dass die Weiterbildungsmaßnahmen in Deutschland rückständig sind. Sie werden zu selten und in zu kleinem Umfang von Firmen angeboten. Vor allem aber kommen sie oft zu spät, wenn also erkannt wird, dass erforderliche Kompetenzen nicht vorhanden sind:

So sind beispielsweise die Inhalte einiger Fortbildungen überholt und der Begriff "Fort"-Bildung falsch gewählt. Begründet wird dieser Zustand in Deutschland oft mit dem Satz: "In Deutschland mahlen die Mühlen langsamer.", was bedeuten soll, dass deutsche Unternehmen nur ungern ihre bisherigen Strukturen und Arbeitsweisen ändern. Erst wenn sie eindeutig erkennen (z.B. "wenn es in Nachbarländern klappt"), dass ihre Struktur überholt ist, werden entsprechende "Fort"-Bildungen angeboten. Eigentlich sind diese Fortbildungen dann jedoch dazu da, nicht den Anschluss zu verlieren und zu anderen ("die es vorgemacht haben") Konkurrenten (oft im Ausland) aufzuschließen.

Ein wenig einschränken müssen wir die Behauptung, dass die Informatik Wissen besonders schnell entwertet. Es ist richtig, dass sich die Informatik wie jede Wissenschaft weiterentwickelt, und dass neue Forschungsergebnisse zu Fortbildungen führen müssen. Jedoch gibt es in der Informatik, wie in allen anderen Wissenschaften auch, gewisse Grundstrukturen, welche ewig erhalten bleiben und sich nicht "entwerten". Trotzdem sollte natürlich ein guter Informatiker immer "auf dem neusten Stand" sein, nur sollte man in diesem Zusammenhang weniger von "Wissen entwerten" reden.

Zustimmen müssen wir, dass sich das Wissen in der Informatik schneller entwickelt als bei anderen Wissenschaften. Dies liegt mit Sicherheit daran, dass die Informatik im Vergleich zum Beispiel zur Mathematik noch in ihren "Kinderschuhen" steckt und sich gerade erst entwickelt. Es muss jedoch hier gewarnt werden: Wissen in der Informatik ist nicht das Kennen der neusten Software. Viele Unternehmen meinen, eine Weiterbildung im Bereich der Informatik vermittelt den Teilnehmern Computerkenntnisse, damit diese später besser Büroarbeiten oder "Hausmeisterarbeiten" (hier der Bezug zur letzten Vorlesung) ausführen können. Informatik ist nach wie vor eine tiefere Wissenschaft, die über Softwareanwendungen und Computerbeherrschung hinausgeht!

Zur fundierteren Stellungnahme haben wir eine weitere Quelle hinzugezogen:

Nach den Ergebnissen des Bundesinstituts für Berufsbildung ist die Anzahl der weiterbildenden Unternehmen in Deutschland im Vergleich zu der Anzahl in anderen

europäischen Ländern relativ gering. Besonders kritisch sieht es in Deutschland mit der Intensität der Weiterbildung aus. Ähnlich wie auch in Großbritannien ist die Stundenzahl pro Teilnehmer in einem Jahr für Weiterbildung vergleichsweise sehr gering. Weiter zeigen die Ergebnisse der BiBb, dass die Kosten für Weiterbildung in Deutschland im Vergleich äußerst hoch sind. Hier wird sicherlich eine Ursache dafür liegen, dass in Deutschland weniger Fortbildungen stattfinden als in anderen Ländern. Zusammenfassen lassen sich die Ergebnisse damit, dass der Grad der beruflichen Weiterbildung in Deutschland im europäischen Vergleich im unteren Drittel aufzufinden ist.

Diese Ergebnisse stützen unsere Ansicht und die Behauptung der Aufgabenstellung. Als Ursachen vermuten wir die hohen Kosten für Fortbildung in Deutschland (wie oben erwähnt) und, wie bereits zuvor angedeutet, die Festigkeit der Strukturen in deutschen Unternehmen: Deutsche Unternehmen sind in den seltensten Fällen offen für Neuerungen und Fortschritt, nicht nur weil diese Veränderungen Zeit, Geld und Arbeit kostet. Deutsche Betriebe haben oft den konservativen Standpunkt, dass alte Verfahrensweisen die besten sind, vielleicht nicht immer die schnellsten, aber zumindest immer die zuverlässigsten.

Eine Folge davon ist die Rückständigkeit im globalen Umfeld. Deutsche Unternehmen haben oft den Ruf, etwas "hinterher" zu sein. Trotzdem müssen wir auch hier einschränkend sagen, dass deutsche Unternehmen im globalen Vergleich trotz der schlechten Weiterbildung ein gutes Ansehen haben. Also ist Weiterbildung zwar wichtig, aber es kommen auch andere Kriterien (z.B. Zuverlässigkeit) bei einer Bewertung hinzu.

Als weitere Einschränkung könnte man einwerfen, dass sich zum Beispiel im Bereich der Lehrerfortbildung in der Vergangenheit in Deutschland einiges getan hat. So finden Lehrerfortbildungen weit häufiger und umfangreicher statt als es in der Vergangenheit der Fall war. Leider zeigen PISA und Co keine positiven Ergebnisse dieser Entwicklung, allerdings sollten deren Ergebnisse auch keine Folge der vermehrten Fortbildungen sein!

weitere Quellen:

Die Ergebnisse des Forschungsberichts 2003 des Bundesinstituts für Berufsbildung
<http://www.bibb.de/>

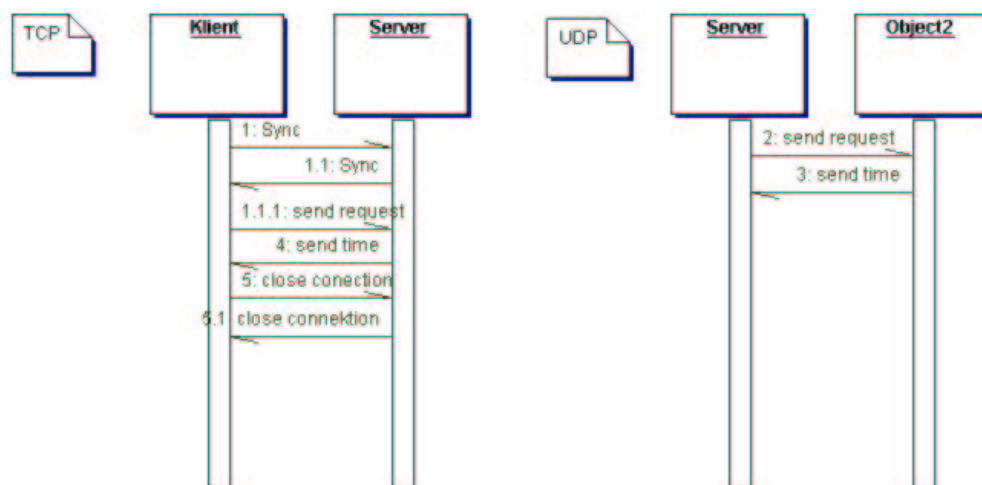
Aufgabe 1)

- a) In vernetzten Umgebungen werden abgestimmte Zeiten für verschiedene Abläufe benötigt. Zum einen bei der gemeinsamen Nutzung von Dateien. Anhand der Zeiten kann erkannt werden, welches die neuste Version der Datei ist. Weiterhin ist es sinnvoll bei Zeitabsprachen unter den Personen im Netz, oder beim Datenaustausch, wenn ein Port nur zu einer bestimmten Zeit offen ist. Ein weiterer Punkt ist der Datenaustausch über ftp, wo z.B. nur neue Dateien ersetzt werden sollen.

Grundsätzliche Ursachen für Zeitdifferenzen sind hauptsächlich die Systemuhren der Einzelsysteme. Da keine Uhr 100% genau geht kommen durch kleine Unregelmäßigkeiten oft große Differenzen zu Stande, oder die Batterien fallen aus, wodurch die Uhr komplett stehen bleibt. Weitere Gründe sind im Netz, das der Datentransport mit leichter Zeitverzögerung statt findet.

- b) *Diese Aufgabe ist eigentlich nicht lösbar, da die Maschinen untereinander nicht kommunizieren können. Nach der Definition von Kommunikation ist eine solche bei gegeben, wenn die Antworten beider Parteien nicht vorhersagbar sind. Also eine Kommunikation kann Typischerweise nur zwischen Menschen statt finden. In diesem Fall wäre das Wort Interaktion das richtige.*

Im RFC wird als mögliche Lösung ein Zeitabgleich angegeben, bei denen die Klienten bei einem Server die richtige Zeit abfragen und diese dann



benutzen.

Weiterhin wird im Singular gesprochen, also welche der beiden Möglichkeiten der Interaktion ist gemeint(UDP oder TCP)?

- c) UDP ist besser für kleine Datenmengen, weil nicht erst eine Verbindung aufgebaut werden muss (paketorientiertes Protokoll), während TCP bei größeren Datenmengen die bessere Wahl ist, da dabei eine Garantie besteht, das die Daten auch ankommen (verbindungsorientiertes Protokoll). Dafür

muß mit den Syn-Paketen am Anfang die Verbindung aufgebaut und am Ende mit Fin-Paketen wieder abgebaut werden.

Für DNS, oder gerade Timeserver-Anfragen ist das UDP Protokoll die bessere Wahl, da sonst der Datenoverhead zu groß wird. Wenn da die Daten halt nicht ankommen fragt man halt nochmal nach. Bei größeren Datenmengen ist das TCP Protokoll besser, da es dabei schlimmer ist, wenn Daten, oder auch nur Teile der Daten verloren gehen. In diesem Beispiel: UDP Ein Paket an den Server. Eines zurück, fertig! TCP Syn-paket zum Server, Antwort des Servers. Dann wie beim UDP und zum Schluß nochmal zwei zum Verbindungsabbau.

4. d) Bei dem UDP Protokoll kann es bei der Internetttelefonie zu Verlusten des Gesprächs kommen, da die Daten verloren gehen können. Dafür ist der Zeitverlust dabei geringer, da nicht bei jedem gesagten eine Verbindung auf- und wieder abgebaut werden muss. So kann es aber zu Mißverständnissen zwischen den Gesprächspartnern kommen. Beim TCP dagegen kann das nicht passieren, aber dafür ist dort ein größerer Zeitunterschied, bis der Hörer das Gesagte zu Ohren bekommt. Also muß man mit langen Wartezeiten rechnen, bis man eine Antwort bekommt.

5. e) Server: Höre auf Port 37

Empfange leeres Paket

Sende Zeit

Klient: Sende leeres Paket auf Port 37

Empfange Zeit

Habe kein Manual zu Python gefunden indem steht, wie man an Ports kommt und damit noch wie mit Python gearbeitet. Deswegen hab ich keine Ahnung wie das geht.

Aufgabe 2)

Durch den verbreiteten Einsatz von Informationssystemen wurde es notwendig die Daten möglichst effizient vielen Nutzern zugänglich zu machen. Daher entstanden größere und kleinere Netze, wie z.B.: das Internet. Durch die unterschiedlichen Informatiksysteme, die untereinander verbunden wurden, entstanden eine Vielzahl von Schwierigkeiten und Problemen, die bewältigt werden mussten. Zum Beispiel das Problem der Zeitdifferenzen zwischen den Systemen.

Aufgabe 3)

1. a) An dieser Stelle muß nichts besonderes besprochen worden sein. Da bei den Schülern, meiner Meinung nach, das Grundwissen vorhanden sein sollte, wozu Zeiten wichtig sind, aber es könnte so etwas ähnliches mal im Unterrichtsgespräch vorgekommen sein.
2. b) Die Schüler müssen im Unterricht verschiedene Formen der Diagrammdarstellung für Systemabläufe gelernt haben, damit sie diese auch anwen-

den können.

3. c) Als wichtigster Punkt muß thematisiert worden sein, wozu Protokolle überhaupt gut sind, und wie sie Arbeiten. Anhand des RTF finde ich, ist es nicht möglich die Unterschiede der Protokolle zu erkennen und auch die Idee der Protokolle wird nicht klar.
4. d) Weiterhin muß dem Schüler klar sein, wie die Internetttelefonie geht, da er sonst die Probleme nicht erkennen kann.
5. e) Der Schüler muß im vorhergegangenen Unterricht die Sprache Python kennengelernt haben, wissen, wie Algorithmen aussehen und wie Ports aus der Programmiersprache her angesprochen werden und auch Daten gesendet werden. Ich finde aber, man kann im Stess schnell überlesen, welcher der Algorithmen jetzt rtf868 entspricht.

Aufgabe 4)

Ich sehe darin einige Schwierigkeiten. Man sollte zwar schon voraussetzen können, dass die Schüler Englisch können, aber da dies ein anderes Fach ist, sollte ein Defizit in der Sprache nicht dazu führen, dass die Aufgaben nicht bearbeitet werden können. In genauem Bezug zu diesem Text, würde ich sagen, dass es ok ist. Er ist nicht beonders lang und auch nicht besonders schwierig. Weiterhin ist er nicht für alle Aufgaben relevant, sondern nur für einen Teil. Der wichtigste Grund ist aber, dass in der Informatik viele Dinge in Englisch sind, so dass Englisch im weitesten Sinne auch zur Informatik gehört.

Aufgabe 1)

Ich würde folgende Punkteskala verwenden: a) 5P, b) 5P, c) 10P, d) 10P, e) 20P Dies mache ich aus folgendem Grund. Die ersten beiden Aufgaben, sind hauptsächlich in den Anforderungsbereich 1 (anwenden von Wissen) einzuordnen. Die zweiten beiden eher in Anwendungsbereich 2 (Bewerten, ect.) und der letzte in den Anwendungsbereich 3 (problemlösendes Denken).

1. a) Lösungsskizze:

- benötigt für verschiedene Abläufe
- Zum einen bei der gemeinsamen Nutzung von Dateien.(die neuste Version)
- Zeitabsprachen unter den Personen im Netz(Datenaustausch)
- Datenaustausch über ftp, wo z.B. nur neue Dateien ersetzt werden sollen.

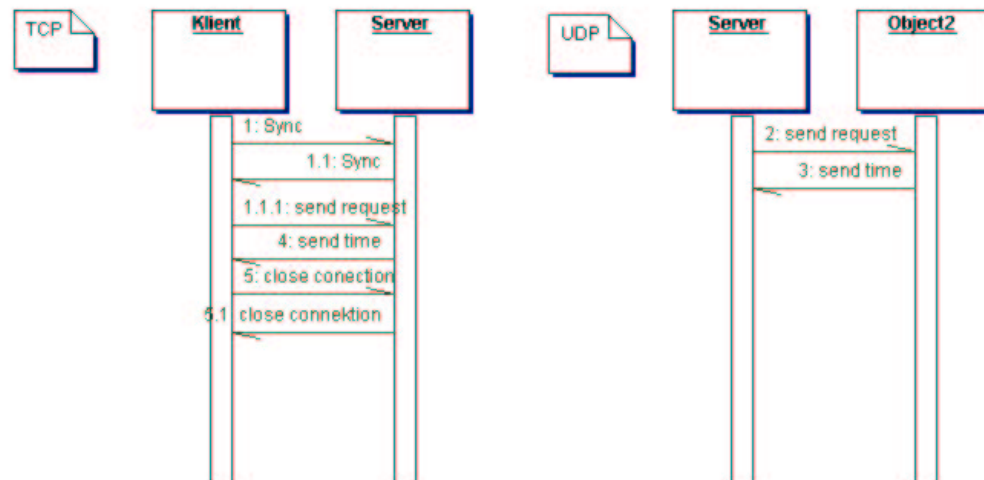
-Grundsätzliche Ursachen für Zeitdifferenzen sind hauptsächlich die Systemuhren der Einzelsysteme. (Unregelmäßigkeiten)

-Falsche Uhrzeit

-Verzögerung beim Datentransport

Soweit die Punkte, die für volle Punktzahl genannt werden sollten. Bei den Antworten der Schüler muß man dann halt sehen, in wie weit sie den Punkt getroffen haben. Diese Aufgabe ist aus dem ersten Anforderungsbereich, da nur Wissen abgefragt wird, was vorher im Unterricht besprochen wurde.

2. b)-Zeitabgleich (Klients fragen bei einem Timeserver die Zeit ab und verwenden diese.)



Hier würde ich für die Angabe der möglichen Lösung 1 Punkt geben und für die richtige Zeichnung der beiden Protokolle die 4 Punkte. Diese Zeichnung fällt schon ein wenig in den Bereich 2, da hier der Schüler selber überlegen muss, wie er es am besten darstellt. Deswegen auch mehr Punkte dafür.

3. c) -UDP für kleine Datenmengen, kein Verbindungsaufbau (paketorientiertes Protokoll)
 TCP bei größeren Datenmengen, Garantie, das die Daten ankommen (verbindungsorientiertes Protokoll).
 -Dafür muß mit den Syn-Paketen am Anfang die Verbindung aufgebaut und am Ende mit Fin-Paketen wieder abgebaut werden.
 -Für DNS, oder gerade Timeserver-Anfragen ist das UDP Protokoll die bessere Wahl (kein Datenoverhead)
 -bei Datenverlust Zweitanfrage
 -bei größeren Datenmengen ist das TCP Protokoll
 -Datenverlust ist hier nicht zu verantworten z.B.: Downloads
 -In diesem Beispiel:
 -UDP 2 Pakete (Anfrage, Daten)
 -TCP 6 Pakete (Syn-Klient, Syn-Server, Anfrage, Daten, Ende Klient, Ende Server)
 Hier würde ich wie folgt verteilen: je zwei Punkte für die richtige Einordnung der Protokolle in ihre Leistung. Für die verschiedenen Einsatzmöglichkeiten der Protokolle und einem Beispiel dafür auch 2 Punkte (1 für Beschreibung und 1 fürs Beispiel) für die Aufschlüsselung der Übertragung auch nochmal zwei. Dies ist eindeutig im Zweiten Bereich zu finden. z.B.: selbständiges Auswählen von Beispielen ect.
4. d) -UDP Protokoll: Verlust von Gesprächsanteilen durch Datenverlust
 -Dafür geringerer Zeitverlust (Nicht immer neuer Verbindungsaufbau)
 -Probleme beim Verständnis des Gegenübers durch Verluste)
 -TCP Protokoll: Lange Wartezeit, bei schlechter Verbindung
 -Möglicherweise große Verzögerung, bis der Gesprächspartner das Gespräch hört
 -Dafür kein Datenverlust
 Für die jeweiligen Vor und Nachteile der beiden Protokolle würde ich je 5 Punkte geben. Die höhere Punktzahl für die Antworten hier kommen dadurch, das der Schüler hier nicht reproduzieren soll, sondern selbst auf diese Fälle kommen soll. Dies ist Aufgabenbereich 2 und auch teilweise schon drei, durch das problemlösenden Denken.
5. e) Server: Höre auf Port 37
 Empfange leeres Paket
 Sende Zeit
 Klient: Sende leeres Paket auf Port 37
 Empfange Zeit
 Hier würde ich 2 Punkte für den verbalen Algorithmus geben. (so wie er oben stehn, hoffe er ist richtig :-)) In dem Programm ist es mit der Punktvergabe schon komplizierter. Für jenden der beiden Programme würde ich 9 Punkte geben, wenn sie komplett richtig sind. Punkte würde ich folgend aufschlüsseln: 1 Punkt für die richtigen Importanweisungen, 1 Punkt für den richtigen Port, 2 Punkte für den richtigen Algorithmus.

Die anderen 5 Punkte sind schwer zu verteilen. Diese würde ich nach dem Gesichtspunkt, wie richtig ist das Programm gecodet geben. Dies ist eindeutig dem Aufgabenbereich 3 zuzuordnen.

Aufgabe 2)

Ich denke alleine aus dem Zurückfallen im Zeitablauf, worunter ich verstehe, das die technologische Leistungsfähigkeit Deutschlands gegenüber anderen Staaten zurückfällt, läßt sich nicht schließen, dass es nur durch die geringeren Vorbildungen der Firmen passiert. Dies ist sicherlich ein großer Grund, aber nicht der alleinige. Weiterhin ist an der Misere sicherlich auch die Abwanderung der Firmen ins Ausland schuld. Nichts desto trotz finde ich aber, dass die Behauptung nicht ganz falsch ist. Wie auch aus der Quelle deutlich wird, sparen die Firmen an allen Ecken und Kanten und da sie von der Weiterbildung im ersten Moment nichts haben, geben sie dafür kein Geld aus. Daraus folgt, dass aber in der folgenden Zeit die Firmen nicht mehr auf dem neustem Stand sind, wodurch sie im Endeffekt etwas verlieren. Sie denken halt nicht zukunftsbetont.

Gerade im Bereich der Informatik ist dies "tötlich". Ich kenne kaum einen Bereich in dem die Entwicklung so schnell geht. Dies liegt zum einen daran, das sie noch relativ jung ist. Andere Bereiche, wie die Mathematik oder Physik, haben sich schon gefestigt, und es sind dort keine weiteren bahnbrechende Neuerungen zu erwarten. Dies ist in der Informatik anders. Dort ist es fast so, dass das, was man an der Uni lernt schon fast veraltet ist, wenn man seinen Hochschulabschluß hat.

Deswegen ist es im IT-Bereich sehr kurzsichtig, keine Weiterbildungen zu bezahlen, wodurch natürlich der Technologiestandort Deutschland immer weiter zurückfällte, wenn in anderen Ländern der Fehler nicht gemacht wird.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 6

In der Übung am 12.6.2003 ist die Installation einer schultypischen Server-Klienten-Infrastruktur geplant. Als Server soll der c't/ODS-Kommunikationsserver zum Einsatz kommen. Lesen Sie zur Vorbereitung die Dokumentation¹ inkl. der Anbindung von Linux-Klienten.

1. Aufgabe (2 Punkte)

Welche Angaben benötigen Sie im Allgemeinen für die erfolgreiche Installation eines Betriebssystems?

2. Aufgabe (3 Punkte)

Skizzieren Sie den Ablauf einer Betriebssysteminstallation.

3. Aufgabe (3 Punkte)

Welche Schritte sind für die Integration eines Linuxklienten erforderlich?

4. Aufgabe (2 Punkte)

Welche Schritte sind notwendig, um einen Internet-Browser und E-Mail-Klienten einzubinden?

Abgabe: 6.6.2003

Besprechung: 12.6.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹Offizielle Heimatseite des c't/ODS-Kommunikationsservers <http://www.heise.de/ct/schan/>
Aktuelle Seite für den c't/ODS-Kommunikationsserver: <http://www.arktur-schule.de/>

Images für CDs (Version 3.3 und Beta für Version 4) befinden sich unter:
<ftp://marvin.sn.schule.de/linux/Image/>

Hinweis: Unter Linux kann eine CD mit dem folgenden Kommando in das normale Dateisystem eingebunden werden:

```
mount -o loop /pfad/zum/image.iso /mnt
```

Übungsblatt 6

1. Aufgabe – Angaben

Hardware	
CPU + Speicher	CPU-Typ RAM
SCSI	Controller Port-Adr. IRQ (Interrupt ReQuest)
Festplatte	Typ : (E)IDE / SCSI Größe Anschluss
CD-ROM	Typ Anschluss
Netzwerkkarte	Typ Port-Adr. IRQ
ISDN-Karte	Typ Port-Adr. IRQ Mem
Parallel-Port	Port-Adr. IRQ
COM-Port	Port-Adr. IRQ

2. Aufgabe – Ablauf einer Betriebssysteminstallation

- Funktionstest des Prozessors, evtl. der Grafikkarte) und Memory Test (Ramtest)
- Prüfung und Zuweisung der Eingebauten Hardwarekomponenten (Plug & Play Karten, CD-ROM, Festplatten, Schnittstellen, evtl. SCSI BIOS, etc.) im BIOS
- Lokalisieren der bootfähigen Partition der Festplatte
- Festlegen, dass der Server von der CD-ROM, bzw. Diskette bootet
- Die CD ins CD-ROM-Laufwerk einlegen (oder die Startdiskette)
- Den Rechner mit RESET starten bzw. den Rechner einschalten. Beim Start erscheint ein Begrüßungsbildschirm und die Meldung „Loading...“
- Für Linux: Der Rechner startet ein einfaches Linux und versucht die CD einzuklinken. Wenn das erfolgreich war, erscheint das Menü: „ODS-Server_Installation“ sowie die Auswahl zwischen: „alles“ mit „alles neu aufbauen“ und „boot“ mit „wieder bootfähig machen“
- a) Für die Erstinstallation: „alles“ auswählen. Die Festplatte wird neu partitioniert und formatiert. Dann werden die Pakete im Slackware-Format auf der Festplatte ausgepackt und eingerichtet.
- b) Falls vorhanden: Setup-Datei, (Optionen – Aktualisieren, nicht neu installieren)
- a) Pakete und Software werden kopiert
- Der Boot-Lader LILO (für Linux) wird eingerichtet. Ist der Vorgang abgeschlossen, kommt die Meldung: „CD-ROM / Diskette entnehmen und den Server neu starten“
- RESET oder den Rechner neu starten und den Server booten. Es erscheint die Meldung „Lilo“ und darauf „boot Kommsver“ und die Ersteinrichtung beginnt.

Ersteinrichtung

- Zuerst erscheint der erste Startbildschirm
- Dem Nutzer Sysadm ein Passwort eingeben.
- Den Domainnamen eingeben
- Einrichtung von Plug & Play – Karten. Wenn man solche Karten hat, sollte man mit „Ja“ antworten.
- Hat der Server PnP-Karten gefunden, werden diese angezeigt.
- Die erste Netzwerkkarte wird eingerichtet. Dafür muss man den entsprechenden Kartentyp aus der Liste auswählen. Je nach dem Typ der Karte muss man Port-Adresse, IRQ eingeben.
- Der Server richtet den Proxy-Cache ein. Dann erfolgt der erste richtige Start des Servers.
- Es folgt die Anmeldung als sysadm mit dem gerade vergebenen Passwort. Die letzten Dienste werden initialisiert. Der Server ist zur Arbeit bereit. Es erscheint das Hauptmenü des Servers.

Der Zugang für root

- Im Menü „Anwender verwalten“ „Einzel“ und dann „Pass“ klicken. Dort den Punkt „ALLES“ auswählen und den Nutzer „root“ suchen. Gültiges Passwort eintragen.
- Man kann sich an einer anderen Konsole einmal als root anmelden,
- Linux herunterfahren: Linux unterscheidet die Varianten „anhalten“ und „neu starten“. Im Hauptmenü gibt es unter „Nutzen“ den Punkt „Ausschalten“. Dann „Anhalten“ oder „Neustart“ auswählen. Die Tastenkombination [Strg]+[Alt]+[Entf] führt zum Anhalten des Servers. Nach einer Minute kommt die Meldung „System angehalten“. Dann kann man den Rechner ausschalten.

3. Aufgabe - Linuxklient

- Einrichten der Netzwerkkarte (Hardware)
- Einrichten des TCP/IP-Protokolls
- Anbindung an Arktur als Fileserver
- Testen der Verbindung
- Nutzen der Verbindung
- Einrichtung weiterer Programme

Für die Anbindung von Linuxklient werden zwei Serverdienste benötigt

- NFS (Network-File-System) – für die Verteilung von Dateien
 - NIS (Network Information System) – für die Verwaltung der Nutzerdaten und ihrer Passwörter
- a) Vorbereitungen an Arktur
- Auf der Serverseite „System verwalten“
 - „Netzdienste (de-)aktivieren“ wählen
 - Es erscheint ein Fenster „Netzdienste einstellen“. Hier „NIS-(YP)-Server Einschalten“ und „NFS-Server Einschalten“ auswählen.
 - Nach dem obligatorischen „AKTIVIEREN“ wird der NFS- und der NIS-Server gestartet.

b) Netzwerkkarte und Protokoll einrichten

- Unter SuSE-Linux dient das grafische YaST2 zur Konfiguration. Bei der Ersteinrichtung startet YaST2 sofort.
- Hier aus der Kategorie „Netzwerk/Basis“ den Punkt „Konfiguration der Netzwerkkarte“ auswählen. Wenn die automatische Hardwareerkennung die Netzwerkkarte bereits gefunden hat, wird diese in der folgenden Liste angezeigt. Danach muss man nur die TCP/IP-Einstellungen vornehmen oder prüfen
- Wenn die Netzwerkkarte nicht erkannt wurde, muss man „Hinzufügen“ wählen.
- Nun erscheint die Auswahl der erkannten Netzwerkschnittstellen. In diesem Fall „Karte aktivieren“ wählen. Wenn die Netzwerkkarte nicht in der Auswahl erscheint, „Karte einrichten“ wählen.

- Den Typ des Netzwerks und die interne Gerätenummer der Karte auswählen.
- „Treibermodul“ angeben. Wenn der Name des Moduls nicht bekannt, dann die Schaltfläche „Treffer Sie eine Auswahl aus der Liste“ wählen.
- Es erscheint eine Liste der Netzwerkkarten-Systeme. Aus dieser Liste muss man einen Typ wählen, der der Hardware am besten entspricht

- Zuletzt muss man die TCP/IP-Einstellungen vornehmen:
 - Am schnellsten: „Automatische Adressvergabe (DHCP) wählen. Dann werden alle weiteren Einstellungen, vom DHCP-Server auf Arktur geholt
 - Alternativ: statische IP-Adresse angeben.
 - „Rechner und Namensserver“ wählen.
 - Im folgendem Fenster den Namen der Arbeitsstation und den Domainnamen (den man in Schulnetz verwendet) angeben
 - In der Liste der Nameserver die IP-Adresse von Arktur angeben – „192.168.0.1“

c) NFS- und NIS-Client einrichten

- Im YaST2 Hauptmenü „Netzwerk/Erweitert“
 - „NFS-Client“ wählen
 Anschließend den NFS-Client konfigurieren. Die Liste der Netzlaufwerke ist zunächst leer. Wählen „Hinzufügen“.
 - Das „Verzeichnis „/home“ von Arktur mit „/home“ auf dem lokalen Client verbinden
- Im YaST2 Hauptmenü „Netzwerk/Erweitert“
 - „NIS-Client“ wählen
- NIS-Domain (die gleiche mit der Schuldomain) und die IP-Adresse (von Arktur: „192.168.0.1“) des NIS-Servers angeben
- YaST testet, ob der angegebenen Server erreichbar und der Dienst bereitgestellt wird. Geöffnete Fenster schließen. YaST trägt alle Angaben in einem Konfigurationslauf ein.
- Den Client neu starten.

d) Netscape Navigator oder Konquerer als Browser einrichten

4. Aufgabe - Internet-Browser und E-Mail-Klient

Installation des Netscape Communicators

- Die verschiedenen Versionen des Netscape Communicators befinden sich vorinstalliert auf dem Server. Im Laufwerk P:, das in der Regel mit „pub“ verbunden ist, liegt unter „software“ das Verzeichnis „netscape“, in dem die einzelne Versionen liegen. Unter Linux wird der Netscape Communicator über den jeweiligen Paketmanager (z.B. YaST bei SuSE) installiert. Dabei wird die benutzerspezifische Konfiguration automatisch im Heimatverzeichnis des Benutzers erstellt.
- Das Heimatverzeichnis des Benutzers muss mit einem Laufwerksbuchstaben verbunden sein (im Windows-Explorer „Extras“ und dann den Punkt „Netzlaufwerk verbinden“ wählen).
- Netscape Communicator starten
- Profil erstellen. Wenn der Communicator gemeinsam mit anderen benutzt wird, kann man mit Hilfe der Profile die Informationen der einzelnen Benutzer voneinander trennen. Jeder Benutzer sollte ein eigenes Profil anlegen und evtl. mit einem Kennwort sichern. Man kann z.B. verschiedene Profile auch für die berufliche und private Nutzung anlegen.
- Im Fenster „Neues Profil anlegen“ „Weiter“ anklicken.
- Benutzername und E-Mail-Adresse eintragen und dann „Weiter“ anklicken. Damit versucht der Profilmanager nun eine Profilkennung zu erzeugen.

- Das Profilverzeichnis auf den Fileserver ins Heimatverzeichnis legen. Dadurch wird erreicht, dass die prej.js (wo die Einstellungen gespeichert sind) immer aus dem Verzeichnis netscape gelesen wird.

Mail-Einstellungen

- Server für ausgehende Mail (SMTP) eingeben: „Arktur“. „Weiter“ klicken.
- Server für eingehende Mail (POP3) einrichten:
 - Den Benutzername für Mail-Server eingeben
 - Server für eingehende Mail eingeben: „Arktur“
 - Art des Mail-Servers auswählen: „POP3“

Dann „Weiter“ anklicken.

-Newseinstellung festlegen:

- Foren-Server eingeben: „Arktur“
- Port eingeben: „119“

Dann „Fertig stellen“ anklicken. Netscape wird versuchen zu starten. Es erscheint eine Fehlermeldung, weil Arktur nicht „Online“ ist und die Einstellungen für den Proxyserver fehlen.

- Im Menü „Bearbeiten“ den letzten Eintrag „Einstellungen“ aufrufen.
- Es erscheint ein Menüfeld, in dem man die Startseite festlegt

Proxy-Einstellungen: Arktur besitzt eine Datei, aus der die automatische Proxy-Konfiguration übernommen werden kann.

- Automatische Proxy-Konfiguration auswählen. Die Adresse ist: „Arktur/netscape.pac“

Cache-Einstellungen korrigieren:

- Festplatten-Cache sollte kleiner werden (aber nicht auf 0 setzen)

Nutzereinstellungen kontrollieren:

- Zum Senden von Mail müssen die folgenden Angaben gemacht werden
 - Name
 - E-Mail-Adresse, etc.

Unter dem Eintrag „Mail-Server“ kann man die Mail-Einstellungen anpassen. Wenn die Liste „Server für eingehende Mail“ leer ist, „Hinzufügen“ wählen und „Arktur“ als Server eintragen. Die Konfiguration mit „OK“ abschließen. Nun sollte Netscape die Startseite des Servers zeigen.

DDI Blatt 6

Aufgabe 1:

- Art und Taktung des Prozessors
- Größe des Arbeitsspeichers
- Festplattengröße und freier Speicherplatz
- Existieren SCSI – Schnittstellen und angeschlossene Geräte? Wenn ja, Angaben über Port und IRQ.
- Gibt es ein CD Laufwerk, wenn ja, wie schnell ist es?
- Wenn es kein CD-Laufwerk in dem vorhandenen System gibt:
Existiert ein Diskettenlaufwerk und gibt es das Betriebssystem auf Disketten?
- Besteht die Möglichkeit, dass Betriebssystem über ein bestehendes Netzwerk zu installieren?
- Existieren für alle im PC vorhandenen Komponenten Treiber für das ausgewählte Betriebssystem?
- Ist die Installationsversion des Betriebssystems mit Seriennummer bzw. Kennnummer vorhanden?
- Angaben über Parallel- und COM- Port, Adresse und IRQ

Aufgabe 2:

- Datensicherung der wichtigen Daten des alten Systems.
- (gegebenenfalls) erstellen einer Startdiskette
- Nach Möglichkeit, Formatierung bzw., wenn nötig, Partitonierung der Festplatte
- Einstellen der Boot- Reihenfolge im BIOS
- Einlegen der CD auf der sich das Betriebssystem befindet und, wenn es sich um keine bootfähige CD handelt, muss eine Bootdiskette mit CD-Laufwerktreiber bzw. Netzwerktreiber verwendet werden.
- Neustart des Rechners
- Start der Installation des Betriebssystems und weiteres Vorgehen nach Anweisungen während des Installationsvorgangs.
- Sollte die Installation nicht von alleine gestartet werden, muss in der Umgebung der Bootdiskette auf das CD-Laufwerk bzw. Netzlaufwerk gewechselt werden und die Installation mit Hilfe der Startdatei gestartet werden.
- Nach Beendigung der Installation sollte die CD und Diskette aus dem jeweiligen Laufwerk entfernt und der PC neu gestartet werden.
- Wurde das Betriebssystem korrekt installiert, müssen nun noch alle nicht erkannten Systemkomponenten mittels entsprechender Treiber im System eingebunden werden.
- Starten des Rechners und damit des neuen Betriebsystems

Aufgabe 3:

- Zuordnung von Nutzer und UID
- UID Bereich ist abhängig von der Funktion des Nutzers
- Zuordnung von Real Name und UID
- Erstellung von Login- Kennung und Passwort
- Angeben des Home Verzeichnisses
- Angeben der Shell
- eventuelles eintragen von Fristen für Passwörter
- Einstellungen über Nutzer Befugnisse bearbeiten

Aufgabe 4:

- eine funktionsfähige Internetverbindung wird benötigt
- der Internet-Browser muss als Installationsversion auf CD oder Diskette vorliegen, ebenso der E-Mail-Klient
- der Internetbrowser kann nun von CD oder Diskette installiert werden; während der Installation muss bei den meisten Browsern die Art der Internetverbindung angegeben werden die für einen Internetzugang mittels des entsprechenden Browsers benutzt werden soll; bei einem Internetzugang über ein Netzwerk wird in der Netzwerkverbindung zum Internetserver der Standardgateway und der Bevorzugte DNS-Server angegeben
- für den E-Mail-Klient wird ein aktives Internetkonto mit z.B. POP3- und SMTP-Server benötigt; nach der Installation des E-Mail-Klient müssen diese Angaben wie auch die korrekte E-Mail-Adresse mit Kennwort bei der Einrichtung eines neuen Kontos eingegeben werden



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 7

1. Aufgabe (6 Punkte)

Der NRW-Lehrplan für das Schulfach Informatik in der Sekundarstufe II empfiehlt die Arbeit mit zwei verschiedenen Programmierparadigmen. Bevorzugt werden bisher imperative und objektorientierte Denk- und Arbeitsweisen vermittelt, aber auch die beiden anderen Paradigmen (deklarativ, funktional) werden im Lehrplan expliziert.

- Charakterisieren Sie knapp die Unterschiede und Gemeinsamkeiten der vier Paradigmen. (2 Punkte)
- Diskutieren Sie Vor- und Nachteile des deklarativen Paradigmas im Hinblick auf eine unterrichtliche Umsetzung. Vergleichen Sie dabei auch mit dem imperativen und objektorientierten Paradigma. (3 Punkte)
- Beurteilen Sie die Eignung des deklarativen Paradigmas für den Anfängerunterricht. (1 Punkt)

Nehmen Sie bei der Beantwortung der Fragen u.a. auch Bezug auf den Lehrplan.
Hinweis: Sie finden möglicherweise in der Diplomarbeit¹ von Fotis Georgatos Anregungen für die Beantwortung der Fragen.

2. Aufgabe (4 Punkte)

Diese Übung folgt dem deklarativen Programmierparadigma.
Gegeben ist ein einfaches Labyrinth und ein Algorithmus, der einen Ausweg aus dem Labyrinth sucht².

- Führen Sie den Algorithmus aus und geben Sie die Bildschirmausgabe ab, in der der Ausgang gefunden wird. (0,5 Punkte)
- Beschreiben Sie umgangssprachlich die Regeln, durch die der Algorithmus realisiert wird. (0,5 Punkte)
- Zeigen Sie auf, wie im deklarativen Programmierparadigma eine Lösung gefunden wird. (1 Punkt)
- Geben Sie eine konkrete Unterrichtsreihe an, in der das Beispiel Bestandteil des Unterrichts ist. (2 Punkte)

Abgabe: 13.6.2003

Besprechung: 26.6.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹URL: <http://www.mikroskosmos.com/DAPE2002/rr.pdf>

²In dem kollaborativen Arbeitsbereich ist die in Python geschriebene deklarative Programmierumgebung sowie der Algorithmus für Sie hinterlegt.

Übungsblatt 7

Aufgabe 1:

a) Um die Unterschiede und die Gemeinsamkeiten der vier Paradigmen am besten ablesen zu können, haben wir uns dazu entschlossen, ein Überblick der Paradigmen in Tabellenform darzustellen:

	Imperativer Ansatz	Objektorientierter Ansatz
Informatikmodelle gewinnen	<ul style="list-style-type: none"> - Computer wird Abfolge von Handlungsanweisungen zur Abarbeitung übergeben - Ein-/Ausgabeprinzip - Top-Down-Methode - Erledigung von Teilaufgaben an Unterprogrammen 	<ul style="list-style-type: none"> - Probleme mit objektorientierter Analyseverfahren modellieren - Objekte führen selbstständig Aufträge aus und interagieren - Realität ist durch Objekte beschrieben - objektorientierte Analyse (OOA) - Objektbeziehungen visualisieren
Daten und Algorithmen abstrahieren	<ul style="list-style-type: none"> - schrittweise Verfeinerung - Bottom-Up-Methode 	<ul style="list-style-type: none"> - objektorientiertes Design (OOD) - Wiederverwendbarkeit und Entwurf neuer Klassen - Vererbung, Polymorphie - Dokumentationstechniken: Schnittstellen von Klassen werden im Protokoll beschrieben; Dienste werden mit Vor- und Nachbedingungen beschrieben
Typische Einsatzbereiche	<ul style="list-style-type: none"> - Softwareprodukte, die nach dem Ein-/Ausgabeprinzip konstruiert sind 	<ul style="list-style-type: none"> - Animation - objektorientierte Betriebssysteme
Lösungskonzepte realisieren und überprüfen	<ul style="list-style-type: none"> - Lösung ist auf Plausibilität zu untersuchen - generelle Zweckmäßigkeit, plangemäße Übersetzung und Güte der Lösungen sind zu überprüfen 	<ul style="list-style-type: none"> - objektorientierte Programmierung (OOP) - ereignisgesteuerte Programmierung - durch Unabhängigkeit der Objekte leicht zu testen - ansonsten Testen von Teilgruppen von Objekten
Algorithmen beurteilen	<ul style="list-style-type: none"> - Untersuchungen sind mit primitiven, imperativ geprägten Ansätzen verknüpft und können an imperativen Programmen gut untersucht werden 	<ul style="list-style-type: none"> - Automatenmodelle mit Objekten leicht zu realisieren - somit können Compiler entwickelt und dabei Sprachkonzepte beurteilt werden
Hard- und Softwaresysteme einordnen	<ul style="list-style-type: none"> - Prozess der Transformation der Daten- und Ablaufstrukturen lässt sich nachvollziehen - gewonnen Erkenntnisse sowie Verständnis für Konstrukte in der imperativen Programmiersprache vertiefen 	<ul style="list-style-type: none"> - Kommunikationsstrukturen - Zusammenwirken der einzelnen Komponenten wird nachvollzogen

	Deklarativer Ansatz	Funktionaler Ansatz
Informatikmodelle gewinnen	<ul style="list-style-type: none"> - in Rechnersystemen ist Wissen gespeichert, so dass sie folgern und passend antworten können - reale Sachzusammenhänge werden analysiert, so dass Wissen als logisches Geflecht erfasst wird und Relationen zwischen Objekten aufgestellt werden 	<ul style="list-style-type: none"> - Verhalten des System stellt sich als Funktion in mehreren Variablen dar - es werden komplexe Probleme in Teilprobleme zerlegt – Sammlung von Funktionen (Umgebung)
Daten und Algorithmen abstrahieren	<ul style="list-style-type: none"> - Wissen beschreibt über Objekte durch Klauseln aus Fakten und Regeln eine Umsetzung in Richtung eines logikorientierten Systems - Datenstruktur Liste - Kopf-Rest-Methode - zum Lösen von Problemklassen Erarbeitung von Suchverfahren 	<ul style="list-style-type: none"> - es werden Grundfunktionen (primitive Ausdrücke) bereitgestellt, welche Baumstruktur besitzen - Kombination - Abstraktion - Aufbau von Bibliotheken - Kopf-Rest-Methode - Anspruchsvolle Aufgaben werden mit Rekursion gelöst - Mechanismus der Bindung von Namen an Werte - Daten als Funktionen, Funktionen als Daten
Typische Einsatzbereiche	<ul style="list-style-type: none"> - Datenbank- und Expertensysteme werden analysiert und bewertet - kleine Modelle werden entwickelt 	<ul style="list-style-type: none"> - künstliche Intelligenz - Textbearbeitung - symbolische Mathematik
Lösungskonzepte realisieren und überprüfen	<ul style="list-style-type: none"> - Unterscheidung zwischen Gehalt und Ablaufsteuerung - Konzentration auf Beschreibung des Problems - Beschränkung auf logische Beziehungen 	<ul style="list-style-type: none"> - Funktionen einer Umgebung können unabhängig voneinander geschrieben und getestet werden - Ausdrücke auswerten (eval) und Funktion auf Argumente anwenden (apply) - Einführung lokaler Variablen, die Zustand eines Objektes darstellen
Algorithmen beurteilen	Dazu steht nichts im Lehrplan	<ul style="list-style-type: none"> - Beschäftigung mit Grammatiken, formalen Sprachen und Automaten - Parser und Compiler lassen sich entwickeln
Hard- und Softwaresysteme einordnen	<ul style="list-style-type: none"> - Untersuchungen zur maschinellen Sprachverarbeitung - Parser, Interpreter und Übersetzer können definiert werden 	<ul style="list-style-type: none"> - Simulation digitaler Schaltungen, deren Verhalten funktional dargestellt wird - signalgesteuerte Prozesse lassen sich mit Streams darstellen

Quelle: NRW-Lehrplan für das Schulfach Informatik Sek 2: Kapitel 2.3.3, S. 27 ff

b)

Wichtig zu erwähnen ist zu Beginn, dass man im Laufe des Informatikunterrichts mindestens einen Paradigmenwechsel vornehmen sollte. Man sollte den Schülerinnen damit verdeutlichen, dass es auch andere Lösungsmöglichkeiten für Probleme gibt, und dass sich, je nach Problemstellung, einige Paradigmen besser eignen als andere.

Das deklarative Paradigma sollte unserer Meinung nur bedingt im Schulunterricht eingesetzt werden. Es überwiegen die Nachteile bei der Betrachtung dieses Ansatzes. Schon gar nicht sollte daran gedacht werden, diesen Ansatz als "Einführung" zu benutzen, d.h. ohne vorherige informatische Bildung direkt dieses Paradigma im Unterricht einzusetzen. Denn dafür ist es viel zu komplex:

Schülerinnen würden Probleme damit haben, Algorithmen zu durchschauen und Problemstellungen zu entschlüsseln. Die Problemstellungen, bei welchen sich das deklarative Paradigma gut anwenden lässt, sind oft nicht sehr praktisch und somit als Einführung ungeeignet.

Der deklarative Ansatz ist unserer Meinung nach auch relativ demotivierend für Schülerinnen, da (wie gesagt) keine praktischen Probleme betrachtet und gelöst werden sondern auf einem komplexen Level gearbeitet wird, bei welchem es oftmals schwierig werden könnte, etwas Gelerntes umzusetzen. Außerdem zeigt das deklarative Paradigma nur einen sehr kleinen Ausschnitt aus der Arbeit der Wissenschaft Informatik, und könnte so "abschreckend" wirken.

Ein weiterer Nachteil liegt darin, dass die Umsetzung eines deklarativen Paradigmas sehr Zeitaufwendig ist. Es sind viele Voraussetzungen notwendig, um mit dem deklarativen Paradigma arbeiten und Problemstellungen lösen zu können. So müssen z.B. Datenstrukturen wie Listen und Bäume vorher durchgenommen werden. Problemstellungen sind außerdem selten "klein" und trivial, sondern werden schnell sehr komplex und verstrickt. Ein kurzes "Kennenlernen" dieses Ansatzes im Unterricht ist also nicht möglich.

Ist eine gute Basis an Vorwissen geschaffen und haben die Schülerinnen einen gewissen Grad an Abstraktionsfähigkeit erreicht, so lässt sich auch der deklarative Ansatz im Schulunterricht einbinden. Schwerpunktmäßig können logische Beziehungen von Problemstellungen aufgestellt und umgesetzt werden. Positiv dabei ist, dass die Schülerinnen so auch einen anderen Ansatz (außer dem (in der Regel) imperativen) kennen lernen und so schon sehen können, welche Teilgebiete innerhalb der Informatik existieren (z.B. künstliche Intelligenz).

Auch Überlegungen in Bezug auf die künstliche Intelligenz und damit verbunden mit ethischen Fragestellungen können innerhalb des deklarativen Paradigmas im Schulunterricht aufgegriffen werden. So lassen sich vor- und Nachteile, Chancen und Gefahren der KI Entwicklung diskutieren.

Zusammenfassend lässt sich sagen, dass sich der deklarative Ansatz nur schwer im Schulunterricht umsetzen lässt, da er schnell zu komplex wird. Wenn, dann sollte dieser möglichst nach dem imperativen- und objektorientierten- Paradigma im Schulunterricht folgen.

Das imperative Paradigma sollte auf alle Fälle im Schulunterricht durchgenommen werden, denn die meisten Programmiersprachen beruhen auf den imperativen Ansatz. Die Algorithmen sind oft einfach gestrickt und gut zu beurteilen. Daher eignet sich dieses Paradigma besonders gut als "Einstieg" in die Programmierung: Es lassen sich einfache Automatenmodelle mit dem Ein- Ausgabe Prinzip realisieren. Die Schüler können so an praktischen Beispielen auch ohne großes Vorwissen in die Informatik eingeführt werden. Die Vorteile bei der schulischen Umsetzung mit diesem Paradigma überwiegen. Nachteile sehen wir lediglich darin, dass dieser Ansatz irgendwann an seine Grenzen stößt, dass es also Problemstellungen gibt, bei denen dieses Paradigma nicht den besten Lösungsweg gibt. Hier sollte möglichst ein Paradigmenwechsel vorgenommen werden (zum Beispiel zum objektorientierten Paradigma).

Das objektorientierte Paradigma sollte möglichst auch im Unterricht auftauchen, da dieses eine sinnvolle Erweiterung zum imperativen Paradigma ist. So lassen sich Dinge mit dem objektorientierten Paradigma einfacher bearbeiten als mit dem imperativen. Man kann Problemstellungen einfacher grafisch darstellen und somit im Unterricht besser skizzieren. Ein Automatenmodell, welches zuvor evtl. mit dem

imperativen Paradigma besprochen wurde, kann so mit dem objektorientierten Ansatz erneut gelöst werden. Unterschiede, Vor- und Nachteile werden so besonders schnell zwischen den beiden Paradigmen erkenntlich. Des Weiteren ist die Objektorientierung ein wichtiges Thema in der Informatik. Gerade in letzter Zeit hat die Objektorientierung an Gewicht gewonnen. Es ist daher hilfreich, diesen Ansatz schon einmal in der Schule gesehen zu haben, bevor man im Informatikstudium darauf stößt. Problematisch wird dieser Ansatz nur, wenn man ihn als Einführung gedacht hat. Unserer Meinung nach lässt sich die Objektorientierung gut im Informatikunterricht der Schule einbinden, jedoch sollte eine gewisse Grundkenntnis vorhanden sein. Man sollte also schon früher einmal einfache Lösungen für ähnliche Probleme mit dem imperativen Paradigma besprochen haben. So sollte man bspw. in den ersten zwei Jahren der Mittelstufe im Informatikunterricht mit dem imperativen Paradigma gearbeitet haben, bevor man in der Oberstufe dann bedenkenlos mit dem objektorientierten weiterarbeiten kann. Andernfalls kann das objektorientierte Paradigma schnell abstrakt und zu komplex wirken.

c) Wir sind der Meinung, dass sich das deklarative bzw. wissensbasierte Paradigma im Allgemeinen nicht sehr gut für den Anfängerunterricht eignet. Wir denken, dass die SchülerInnen im Anfängerunterricht große Probleme damit haben, sich vorzustellen, dass Wissen in Rechnersystemen gespeichert ist und diese in der Lage sind Folgerungen zu ziehen und passend auf Fragen zu antworten. Da nach dem wissensbasierten Ansatz die SchülerInnen im Unterricht Sachzusammenhänge analysieren, vereinfachen und beschreiben müssen, denken wir, dass dafür ein gewisses Grundwissen vorhanden sein muss, so dass sich dies noch nicht für den Anfängerunterricht eignet. Zudem sollen die SchülerInnen Relationen zwischen Objekten eines Problemraumes aufstellen; hierfür sind z.B. Vorkenntnisse über Objekte und deren Beziehungen unbedingt notwendig.

Um Klauseln und Formulierungen von Anfragen erarbeiten zu können, müssen Kenntnisse über Definitionen von z.B. Wenn-dann-Beziehungen, Konjunktionen, Disjunktionen usw. vorhanden sein. Diese Definitionen sollten im Anfängerunterricht besprochen werden um im weiterführenden Unterricht Klauseln darüber aufstellen zu können.

Auch sind wir der Meinung, dass Rekursion, die im deklarativen Ansatz von Bedeutung ist, nicht für den Anfängerunterricht geeignet ist.

Im Allgemeinen denken wir, dass der wissensbasierte Ansatz eine hohe Abstraktionsfähigkeit der SchülerInnen voraussetzt, die im Anfängerunterricht noch nicht vorhanden ist.

Aufgabe 2:

d)

Der Entwurf eines solchen Algorithmus´ steht offensichtlich im Zusammenhang mit dem Erlernen einer Programmiersprache, in diesem Fall Python. Zunächst müssen hier die entscheidenden Befehle erklärt und an ersten einfachen Beispielen bzw. Algorithmen geübt worden sein, bevor man sich an ein etwas komplexeres Thema heranwagen sollte.

Allerdings ist der Entwurf eines Labyrinths ein erster sehr praktischer Teil des Informatikunterrichts. Es wird nicht nur einfach „trocken“ etwas gelernt oder programmiert, sondern es entsteht bei den SchülerInnen ein erster Eindruck, wozu man nun durch das Programmieren in der Lage ist. Somit ist für uns die

Unterrichtsreihe „Prädikatives Modellieren“ sehr gut dafür geeignet, dieses Thema zu beleuchten:

Nachdem das Wissen der SchülerInnen durch elementare Bestandteile im Bereich der Programmierung gefestigt worden ist, eignet sich das Prädikative Modellieren geradezu hervorragend, die gewonnenen Kenntnisse in diesem interaktiven Beispiel umzusetzen.

Des Weiteren kann man aber auch noch komplexere Unterrichtsreihen durchführen, in denen das Beispiel des Labyrinths vorkommen kann:

So ist es möglich, im weiterführenden Sinne, den Begriff der Rekursion mit diesem Beispiel zu verbinden. Die SchülerInnen könnten den Algorithmus modifizieren, so dass er rekursiv arbeitet, jedoch den gleichen Effekt hat bzw. dasselbe Ergebnis liefert, wie die erste Lösung. Dieses Thema wäre auch sehr gut für die Oberstufe geeignet.

Alles in Allem finden wir dieses Beispiel sehr gut, da es den SchülerInnen eine erste, eigene Programmierungsmöglichkeit bietet und es den meisten sehr viel Spaß bereitet (vgl. Humbert, Ludger: Kollaboratives Lernen - Gruppenarbeit im Informatikunterricht.).



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 8

Funktionales Programmieren mit Scheme

1. Aufgabe:

1 Punkt

Die funktionale Programmiersprache Scheme zeichnet sich dadurch aus, dass sie im Kern aus nur wenigen Sprachkonstrukten besteht, die in relativ kurzer Zeit erlernt werden können. Die Sprache besteht aus vollständig geklammerten Ausdrücken in Präfix-Notation .

Bsp.: $\frac{x^2 A 1}{2}$ wird in Scheme geschrieben als `(/ (+ (expt x 2) 1) 2)`

Schreiben Sie folgende Ausdrücke in Präfix-Notation:

- a) $9\frac{3}{4}$
- b) $x^2 A 20$
- c) $\frac{1}{2} x^2 A 10$
- d) $2B\frac{1}{x}$

2. Aufgabe:

3 Punkte

Funktionale Programme werden nach folgendem Muster definiert:

```
(define (<programmname> <attr1> ... <attrn>)  
      <ein-Ausdruck>)
```

Nach dem Schlüsselwort `define` folgt der Programmname und eine Menge von n ($n \geq 1$) Eingabevariablen (Funktionsdeklaration) sowie das auszuführende Programm (Ausdruck). Beachten Sie die Klammerung.

Bsp.: Das Programm eines Euro-Umrechners kann folgendermaßen geschrieben werden:

```
(define (euro->dm euro)  
      (* 1.95583 euro))
```

- a) Schreiben Sie in Scheme einen Euro-Umrechner für D-Mark in Euro.
- b) Schreiben Sie Programme für die Umrechnung von Celsius in Fahrenheit und umgekehrt. (Hinweis: Die Umrechnungsformeln sind: $F = 9/5 * C + 32$ und $C = 5/9 * (F - 32)$)

3. Aufgabe:**3 Punkte**

Die Funktionale Programmierung ist einer mathematischen Notation sehr ähnlich.

a) Übersetzen Sie das folgende Scheme-Programm in eine mathematische Notation.

```
(define (fib x)
  (if (<= x 1)
      1
      (+ (fib (- x 1))
         (fib (- x 2)))))
```

b) Übersetzen Sie den Euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers in ein Scheme-Programm.

ggt: $\{D\}$

$$\text{ggt} = \begin{cases} a, & \text{falls } a=b \\ \text{ggt}(a-b, b), & \text{falls } a>b \\ \text{ggt}(a, b-a), & \text{falls } a<b \end{cases}$$

4. Aufgabe:**3 Punkte**

Die Türme von Hanoi

Gegeben seien n Scheiben unterschiedlichen Durchmessers. Die Scheiben seien der Größe nach zu einem Turm geschichtet, die größte zuunterst. Dann lautet die Aufgabe, den Turm vom gegebenen Platz a auf einen anderen Platz b zu verlegen. Als Zwischenablage steht ein dritter Platz c zur Verfügung. Es darf stets nur die oberste Scheibe eines Turms bewegt werden. Zu keiner Zeit darf auf einem der drei Plätze eine kleinere Scheibe unter einer größeren liegen.

- Beschreiben Sie einen Algorithmus zur Lösung dieser Aufgabe.
- Welche elementaren Schritte verwenden Sie?
- Wieviele elementare Schritte benötigen Sie?

Abgabe: 20.6.2003

Besprechung: 26.6.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

Hinweis: Sie können zum Testen die interaktive Programmierumgebung DrScheme benutzen (<http://www.drScheme.org>).

Übungsblatt 8

Aufgabe 1:

- a) (+ (/ 3 4) 9)
- b) (+ (expt x 2) 20)
- c) (+ (/ (expt x 2) 2) 10)
- d) (- (2) (/ 1 x))

Aufgabe 2:

- a) (define (dm->euro dm)
 (/ dm 1.95583))
- b) (define (celsius->fahrenheit celsius)
 (+ (* (/ 9 5) celsius) 32))

 (define (fahrenheit->celsius fahrenheit)
 (* (/ 5 9) (- fahrenheit 32))))

Aufgabe 3:

- a)
Das Scheme-Programm beschreibt rekursiv die Fibonacci Funktion:
Fib(x):=Fib(x-1)+Fib(x-2) mit Fib(1) = Fib(0) = 1

b)

```
(define (ggT a b)
  (if (= a b)
      a
      (if (< a b)
          (ggT a (- b a))
          (ggT (- a b) b)
      )
  )
)
```

weiter Quelle:

<http://www.ccs.neu.edu/home/dorai/t-y-scheme/t-y-scheme.html>

benutztes Programm: Dr.Scheme

<http://www.drscheme.org/>

Aufgabe 4:

a)

Verschiebe einen Hanoi-Turm der Höhe n ($n > 0$) von Platz a nach Platz b mit Zwischenablage Platz c:

```
Turmbewegung (n, a, b, c) {      # bewegt Turm mit n Scheiben von Platz a nach Platz b #
    If (n==1) then „Verschiebe Scheibe von Platz a nach Platz b“
    Else begin
        Turmbewegung (n-1, a, c, b);
        „Verschiebe die verbleibende Scheibe nach Platz b“;
        Turmbewegung (n-1, c, b, a);
    end;
}
```

b)

Das entscheidende Element dieses Algorithmus ist die Rekursion; der Algorithmus „Turmbewegung“ ruft sich selbst innerhalb des Algorithmus wieder auf.

Elementare Schritte sind folgende:

- „Verschiebe den Hanoi-Turm der Höhe $n-1$ auf Platz c“; (Rekursion)
- „Verschiebe die verbleibende(unterste) Scheibe nach Platz b“;
- „Verschiebe den Hanoi-Turm der Höhe $n-1$ von Platz c nach Platz b“; (Rekursion)

c) Laufzeitbestimmung:

$$T(n) = 2 \cdot T(n-1) + 1 \text{ mit } T(1) = 1$$

Einsetzen ergibt:

$$\begin{aligned} T(n) &= 2 \cdot T(n-1) + 1 \\ &= 2^2 \cdot T(n-2) + (1+2) \\ &= 2^3 \cdot T(n-3) + (1+2+4) \\ &= 2^k \cdot T(n-k) + (1+2+4+\dots+2^{k-1}) \end{aligned}$$

Mit $k=n-1$ ergibt sich:

$$\begin{aligned} T(n) &= 1+2+4+\dots+2^{n-1} \\ &= 2^n - 1 \\ &= T(2^n) \end{aligned}$$

Also ist die Laufzeit des Algorithmus **$T(2^n)$** .



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 9

Das 7. Übungsblatt beschäftigte sich mit dem deklarativen Paradigma und einer deklarativen Lösung eines Labyrinthbeispiels. In diesem Übungsblatt wird eine funktionale Lösung desselben Beispiels erarbeitet. Verwenden Sie dazu bitte die interaktive Entwicklungsumgebung DrScheme¹.

Im Anhang dieses Übungsblatts ist eine funktionale Lösung des Labyrinthbeispiels und ein Schema zum funktionalen Problemlösen angegeben. Eine elektronische Fassung befindet sich zusätzlich im BSCW-Bereich.

1. Aufgabe: (1 Punkt)

Starten Sie das Labyrinthbeispiel in DrScheme (Sprache „Fortgeschrittene/r“) mit (`tour 'rein 'raus labyrinth-small`). Geben Sie die Ausgabe an.

2. Aufgabe: (6 Punkte)

Bei der Entwicklung einer Problemlösung leisten Schemata eine wertvolle Hilfe. Im Anhang des Skripts finden Sie für verschiedene Paradigmen Schemata für den unterrichtlichen Einsatz. Für das Labyrinthbeispiel ist im Anhang dieses Übungsblatts ein unvollständiges „Entwurfsrezept“ angegeben, das dem Vorgehensmodell von Felleisen et al.: How to design Computer Programs² folgt. Vervollständigen Sie es, indem Sie

- a) für die 2. Phase Beschreibungen für die Funktionen `weg?` und `tour` erstellen, (2 Punkte)
- b) Beispiele angeben, die die Ein- und Ausgabebeziehungen charakterisieren, (1 Punkt)
- c) in natürlicher Sprache die Funktionen beschreiben, die den Algorithmus darstellen (2 Punkte)
- d) den Algorithmus und die Funktionen testen. (1 Punkt)
Falls Sie Fehler im Algorithmus finden, korrigieren Sie sie bitte. (5 Sonderpunkte !)

Geben Sie auch den um die entsprechenden Kommentarzeilen erweiterten Algorithmus ab.

3. Aufgabe: (3 Punkte)

Geben Sie an, wie Sie das Beispiel in den Unterricht der Sekundarstufe II einbetten können. (Vorbedingungen, Lernziele, Dauer, Besonderheiten etc.)

Abgabe: 27.6.2003

Besprechung: 3.7.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹ DrScheme ist kostenlos unter <http://www.drscheme.org> erhältlich

² <http://www.htdp.org/2002-09-22/Book/curriculum-Z-H-2.html>

Anhang

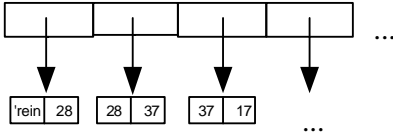
Labyrinthbeispiel-funtional (V0.2)

```
(define labyrinth-small
  (list(list 'rein 28)
        (list 28 37)
        (list 37 17)
        (list 37 47)
        (list 47 46)
        (list 46 66)
        (list 46 45)
        (list 45 35)
        (list 35 25)
        (list 35 33)
        (list 33 'raus)))

(define (weg? location maze)
  (if (equal? maze empty)
      false
      (cond [(or (equal? location (first (first maze)))
                  (equal? location (rest (first maze))))
              true]
            [else (weg? location (rest maze))])))

(define (remove-location a maze)
  (if (equal? maze empty)
      empty
      (if (equal? a (first (first maze)))
          (rest maze)
          (cons (first maze) (remove-location a (rest maze))))))

(define (tour start ziel maze)
  (if (equal? start ziel)
      (display "geschafft!")
      (if (weg? start maze)
          (begin
             (display start)
             (display " ")
             (tour (first (rest (first maze)))
                  ziel
                  (remove-location start maze))
             (tour start ziel (remove-location start maze))
             )
          (display " "))))
```

Phase	Ziel	Aktivität
Problemanalyse & Datendefinition	Formulieren einer Datenstruktur	<p>Es soll eine Wegsuche durch ein Labyrinth realisiert werden. Die Datenstruktur des Labyrinths soll derjenigen des deklarativen Beispiels entsprechen. Als Grundbausteine werden Listen eingesetzt.</p> <p>Labyrinth:</p>  <pre>(list (list 'rein 28) (list 28 37) (list 37 17) (list ...) ...)</pre>
Contract, Purpose & Effect Statements, Header	Benennen der Funktionen, Spezifizieren der Klassen der Ein- und Ausgaben, Beschreibung der beabsichtigten Funktionsweise, Formulieren der Header	<p>Namen für die Funktionen festlegen, ihre Ein- und Ausgaben festlegen, ihr Einsatzgebiet und ihre Funktionsweise beschreiben, Header formulieren:</p> <p>remove-location</p> <pre>;; Contract: remove-location : a maze -> maze ;; Purpose: in einem gegebenen Labyrinth „maze“ ;; soll genau ein Wegstück ausgehend von der ;; Position a ausgehend entfernt werden. Die ;; Ausgabe ist ein Labyrinth ohne diese Kante ;; Examples: (remove-location 'rein labyrinth) ;; = ;; (list (list 28 37) (list 37 17) ...)</pre> <p>Definition:</p> <pre>(define (remove-location a maze) (if (equal? maze empty) empty (if (equal? a (first (first maze))) (rest maze) (cons (first maze) (remove-location a (rest maze))))))</pre> <p>Tests:</p> <pre>;; (remove-location 28 labyrinth) ;; expected value: (list (list 'rein 28) (list 37 17) ...)</pre> <p>weg?</p> <pre>;; Contract:</pre> <p>tour</p> <pre>;; Contract:</pre>
Examples	Charakterisieren der Ein- und Ausgabebeziehung durch Beispiele	...
Body	Definieren der Funktionen	Normalerweise: Programmieren der Funktionen Hier: Natürlichsprachliche Beschreibung der Funktionen
Test	Fehler entdecken	Testen des Algorithmus mit verschiedenen Labyrinthen.

Aufgabe 1

Die Ausgabe sah wie folgt aus:

rein 28 37 37 47 46 46 45 35 35 33 geschafft!

Aufgabe 2

a)

weg?

Contract: `weg? : location maze -> boolean`

Purpose: in einem gegebenen Labyrinth "maze" soll herausgefunden werden, ob es von der Position "a" in diesem Labyrinth ein Weg existiert, also ein Wegstück von "a" zu einer anderen Position im Labyrinth. Die Ausgabe ist ein boolean, der den Wert true annimmt, wenn ein solcher Weg existiert und false, wenn kein solcher Weg existiert.

tour

Contract: `tour : start ziel maze -> Ausgabe`

Purpose: in einem gegebenen Labyrinth "maze" soll ein Weg von der Position "start" zur Position "ziel" gefunden werden. Die Ausgabe gibt die einzelnen Positionen an, welche beim durchlaufen des Labyrinthes vom "start" bis zum "ziel" besucht werden. Kann die Position "ziel" erreicht werden, so wird zusätzlich ein "geschafft!" ausgegeben. Kann die Position "ziel" nicht erreicht werden, so wird kein "geschafft" ausgegeben, statt dessen aber die möglichen Wege, die man vom Start aus gehen konnte. Es werden dabei alle möglichen Wege aufgelistet. Direkte Sackgassen (eine Position, von welcher kein Weg weg führt) werden dabei nicht gegangen und aufgelistet.

b)

Examples:

`(weg? 66 labyrinth-small) -> false`

`(weg? 45 labyrinth-small) -> true`

Bei der Eingabe der Parameter 66 als "location" und labyrinth-small als "maze" wird ein false ausgegeben, weil kein Listeneintrag mit (list 66 x) in labyrinth-small vorhanden ist, wobei x für einen beliebigen Wert steht. Der Eintrag (list 46 66) ist zwar vorhanden, stellt aber keinen Weg von dem Standpunkt 66 zu einem anderen dar, sondern lediglich den Weg zum Standpunkt 66 ((list 46 66) ist also eine Sackgasse, weil kein Weg heraus führt).

Bei der Eingabe der Parameter 45 als "location" und labyrinth-small als "maze" wird ein true ausgegeben, weil der Listeneintrag (list 45 35) in labyrinth-small vorhanden ist.

`(tour 37 25 labyrinth-small) -> 37 28 37 28`

`(tour 28 66 labyrinth-small) -> 28`

Hier lässt sich die Ein- Ausgabebeziehung schlecht beschreiben, da offensichtlich fehler im Alg. vorhanden sind, denn (tour 28 66 labyrinth-small) müßte als eine Ausgabe den Weg "28 37 37 47 46 66 geschafft!" liefern.

c)

weg? (location, maze):

WENN "maze" eine leere Liste ist, DANN gib "false" zurück.

SONST: WENN [(das erste Listenelement des ersten Listenelementes der Liste "maze" = "location" ist) oder (die Liste (außer dem ersten Eintrag!) des ersten Listenelementes der Liste "maze" = "location" ist)],

DANN gib "true" zurück

SONST rufe weg? rekursiv auf, indem "location" unverändert übergeben wird und als Liste "maze" die vorher benutzte Liste bis auf den ersten Eintrag (das erste Listenelement) übergeben wird.

weg? geht alle Listenelement der Liste "maze" durch und schaut, ob von einem Listenelement, welches aus zwei Einträgen besteht, der erste Eintrag = "location" ist. Dann wird ein true ausgegeben und die Liste nicht weiter "durchforstet". Gelangt man an das Ende der Liste "maze", so wird ein false ausgegeben.

Unsere Gruppe versteht die letzte Wenn Bedingung nicht: Wie soll

(equal? location (rest (first maze)))

, also "(ist die Liste (außer dem ersten Eintrag!) des ersten Listenelementes der Liste "maze" = "location"?)" jemals "true" werden? (Wie kann man eine Liste mit einer Zahl vergleichen?)

tour (start, ziel, maze)

WENN Start und Ziel gleich sind, DANN gib "geschafft" aus.

SONST: (WENN ein Weg von "start" zu einer anderen Position existiert,

DANN: (gib "start" aus.

gib Freizeichen aus.

rufe tour rekursiv auf,

[wobei "start" nun die Position ist, welche in der Liste von dem ersten Element der Liste "maze" als zweites Element auftaucht (welche also zu ihr hinführt)

"ziel" bleibt unverändert

die Liste "maze" wird dahingehend verändert, daß der erste Listeneintrag, der mit dem Element "start" beginnt, entfernt wird].

rufe tour rekursiv auf,

[wobei "start" und "ziel" unverändert bleiben, jedoch das erste Element aus der Liste "maze" entfernt wird, welches als erstes Element "start" beinhalten]

)

gib Freizeichen aus.)

tour prüft zunächst, ob "start" und "ziel" identisch sind und gibt gegebenenfalls ein true aus und beendet den Alg.. Anderenfalls wird als nächstes geprüft, ob von der Position "start" ein Weg existiert, mittels unserer Funktion weg?.

Ist dies der Fall, so wird zunächst "start", gefolgt von einem Freizeichen ausgegeben. Es wird daraufhin tour zweimal rekursiv aufgerufen, und zwar, um nun von dem von "start" aus erreichten Knoten (ein solcher existiert, da weg? true lieferte) die tour fortzusetzen und ob zu schauen, ob weitere Wege von "start" aus existieren.

Der zweite rekursive Aufruf von tour ist dabei notwendig, um alle Wege von "start" aus auszugeben. Dabei kann es natürlich von einer Position "start" verschiedene Abzweigungsmöglichkeiten geben. Deshalb der zweite Aufruf.

Der erste sollte klar sein (es wird der Nachfolger besucht und von dieser Position aus weitergegangen). Es liegt hier jedoch ein Fehler vor (Erläuterung später).

Eine natürliche Beschreibung gestaltet sich hier schwierig, da wir es mit Listen in Listen zu tun haben und mit rekursiven Aufrufen, die sich schlecht umgangssprachlich beschreiben lassen. Wir hoffen, es trotzdem geschafft zu haben.

d)

siehe b) und Aufgabe 1

Ein weiteres Labyrinth sollte auch getestet werden, jedoch ist schon jetzt bemerkt, daß Fehler vorhanden sind. Vor weiterem Testen sollte man diese erst beheben.

Unserer Meinung nach ist in der Funktion **weg?** etwas überflüssig (bzw haben wir etwas nicht verstanden?):

```
(define (weg? location maze)
  (if (equal? maze empty)
      false
      (cond [(equal? location (first (first maze))) true]
            [else (weg? location (rest maze))])))
```

....müßte genau so gut funktionieren (wobei hier evtl noch das cond vereinfacht werden kann durch einfaches "if-then-else")

Testen lieferte gewünschte Ergebnisse.

Es bleiben Fehler bei dem Ausführen von `tour`. Fehlerhaft ist hierbei der erste rekursive Aufruf. Es wird ein veränderter "Start" Parameter übergeben, nämlich `(first (rest (first maze)))`, was aber nicht immer richtig ist. Dies setzt nämlich voraus, daß das erste Listenelement in "maze" den aktuellen Startknoten als erstes Element beinhaltet, was aber zB beim Aufruf von `(tour 28 47 labyrinth-small)` nicht der Fall ist.

Getestet werden kann dies durch:

```
(define (tour start ziel maze)
  (if (equal? start ziel)
      (display "geschafft!")
      (if (weg? start maze)
          (begin
            (display "weg existiert von ")
            (display start)
            (display "! Naechste tour von ")
            (display (first (rest (first maze))))
            (display " nach ")
            (display ziel)
            (display " und tour von ")
            (display start)
            (display " nach ")
            (display ziel)
            (tour (first (rest (first maze))) ziel (remove-location start maze))
            (tour start ziel (remove-location start maze))
          )
          (display " "))))
```

Beim Eingeben von `(tour 28 47 labyrinth-small)` ergibt sich die Ausgabe:

```
"weg existiert von 28! Naechste tour von 28 nach 47 und tour von 28 nach 47"
```

...hier sieht man: die erste rekursive tour ist falsch. Sie müßte von 37 nach 47 gehen!

Zur Lösung unseres Problems schreiben wir eine neue Funktion, welche uns den Nachfolger in "maze" sucht, dessen Vorgänger = "Start" ist.

```
(define (NACH Start maze)
  (if (= (first (first maze)) Start)
      (first(rest(first maze)))
      (NACH Start (rest maze))
  ))
```

...liefert uns zB für `(NACH 28 labyrinth-small)` den Wert "37", wie es also sein sollte.

Zur Erläuterung: Die Funktion NACH geht alle Listeneinträge von "maze" durch, bis es ein Listeneintrag erreicht (der wieder aus einer Liste mit zwei Elementen besteht), dessen erstes Listenelement den Wert "Start" hat. Es wird dann das zweite Listenelement ausgegeben.

Hierbei ist zu beachten, daß in der Liste "maze" nur Zahlen auftauchen sollten. 'rein und 'raus sind also ungünstig und sollten in 0 und 100 umbenannt werden, um lästige Abfragen (ist es ein Integer....) zu vermeiden und die Sache nicht unnötig zu verkomplizieren!

entsprechend verändern wir nun unseren Alg. tour:

```
(define (tour start ziel maze)
  (if (equal? start ziel)
      (display "geschafft! ")
      (if (weg? start maze)
          (begin
            (display start)
            (display " ")
            (tour (NACH start maze) ziel (remove-location start maze))
            (tour start ziel (remove-location start maze))
          )
          (display " "))))
```

Hier wurde nun der zweite rekursive Aufruf korrigiert (dank unserer Funktion).

Eingabe von

(tour 28 17 labyrinth-small)

liefert

"28 37 geschafft! 37 47 46 46 45 35 35 33"

Man könnte nun noch die Ausgabe verbessern: Das nämlich bei einem "geschafft!" die Ausgabe abgebrochen wird (immerhin haben wir ja das Labyrinth verlassen).

Dazu brauchen wir eine Variable und eine neue Funktion:

```
(define ende 0)
```

```
(define (geschafft ziel)
  (begin
    (set! ende 1)
    (display ziel)
    (display " geschafft!")
  ))
```

unserer Alg. tour sähe dann wie folgt aus:

```
(define (tour start ziel maze)
  (if (equal? start ziel)
      (geschafft ziel)
      (if (and (weg? start maze) (= ende 0))
          (begin
            (display start)
            (display " ")
            (tour (NACH start maze) ziel (remove-location start maze))
            (tour start ziel (remove-location start maze))
          )
          (display " "))))
```

Nun Testen wir unsere fertige Version:

Eingabe von (tour 28 17 labyrinth-small) liefert:
" 28 37 17 geschafft!"

Eingabe von (tour 37 35 labyrinth-small) liefert:
37 37 47 46 46 45 35 geschafft!

Eingabe von (tour 46 47 labyrinth-small) liefert:
46 46 45 35 35 33

....sieht doch nett aus. Weitere Verbesserungen lassen wir an dieser Stelle dann bleiben (hat schon lange genug bis hierhin gedauert).

Aufgabe 3

Der prädikative Ansatz, dass Beispiel in den Unterricht einzubetten ist sehr anwendungsorientiert und bietet den SchülerInnen somit gute Modellierungschancen. Allerdings sind für eine erfolgreiche Umsetzung des Themas im Unterricht einige Aspekte zu berücksichtigen:

Als Voraussetzungen für die Implementierung eines Labyrinths müssen die notwendigen System- und Sprachkenntnisse der jeweiligen Programmiersprache integrativ erarbeitet worden sein, so dass die SchülerInnen mit dem notwendigen Wissen ausgestattet sind. Dies kann vorher an theoretischen Beispielen, jedoch durchaus auch schon an ersten, kleineren praktischen Beispielen verdeutlicht worden sein, was unserer Meinung nach sinnvoller ist, da es den SchülerInnen mehr Spaß macht. Wir würden so gar noch einen Schritt weiter gehen und die Notwendigkeit eines praktischen Beispiels im voran gegangenen Unterricht unterstreichen, weil es dann leichter fällt mit dem Labyrinth - Algorithmus umzugehen, da er schon etwas komplexer ist. Somit werden die Vorkenntnisse der SchülerInnen thematisiert.

Die Lernziele, welche hier erreicht werden sollen, sind deutlich erkennbar:

Es wird der Umgang mit Algorithmen geschult und ihre Arbeitsweise untersucht. Dadurch wird nicht nur die praktische Arbeitsweise der SchülerInnen verbessert, sondern auch deren Logik ausgeprägt, was sogar einen allgemein bildenden Charakter hat. Sie sollen dann später so weit fortgeschritten sein, dass sie die Effizienz des Programms beurteilen und mögliche Verbesserungsvorschläge unterbreiten können. Desweiteren entwickeln die SchülerInnen durch eine gezielte, jedoch allgemein gehaltene Anleitung zur Problembehandlung auch selbst diese und adaptieren sie im besten Fall, um sie dann später wieder in einem anderen Kontext zu gebrauchen. Somit ist auch Problembewältigung ein zentrales Lernziel. Allerdings kann es auch noch andere, vielleicht auf den ersten Blick weniger herausstechende Ziele geben:

Da ein solches Beispiel in Teamarbeit durchgeführt werden sollte, wird der Umgang mit MitschülerInnen gefördert, was im besten Fall sogar zu einer Verbesserung des sozialen Klimas in der Klasse führen kann. Auch dies ist ein wichtiges Lernziel.

Das Beispiel selbst sollte unserer Meinung nach in der zwölften Klasse (je nach Wissensstand evtl. erst im zweiten Halbjahr) durchgeführt werden. Es sollte auf eine Dauer von 3 Wochen beschränkt sein, vorausgesetzt es werden 3 Std. Informatikunterricht pro Woche erteilt. Hierbei sind das Zusammentragen der Ergebnisse, sowie notwendige Vorbereitungen auf das konkrete Beispiel mitgerechnet. Um die Effizienz der SchülerInnen zu steigern wäre es wünschenswert, dass in der Woche eine Doppelstunde stattfindet.

Zuletzt möchten wir auf die Besonderheiten des Beispiels zu sprechen kommen und eine eigene Bewertung dessen vornehmen:

Insgesamt sollte das Beispiel im Kontext zu dem Thema „Programmierung und Algorithmenentwurf“ stehen und dann auch in den Unterricht der Sekundarstufe II eingebettet werden. Der Entwurf eines Algorithmus´ für ein Labyrinth ist ein sehr interaktives Beispiel, was sich bei den SchülerInnen meistens großer Beliebtheit erfreut. Desweiteren bereitet es auf die späteren Probleme bei Programmen vor, indem es die Komplexität bei schon „einfachen“ Lösungen im Vergleich zu anderen Programmen verdeutlicht.

Alles in Allem halten wir das Beispiel für sehr sinnvoll, da der anwendungsorientierte Zugang, welcher hier thematisiert wird, von sehr essentieller Natur ist in der Informatik und die Umgehensweise mit Algorithmen vertrauter wird, so dass eine Grundlage für den weiteren Unterricht bis zum Abitur geschaffen wird.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 10

Die funktionale Lösung des Labyrinth-Beispiels von Übungsblatt 9 soll nun in Python überführt werden. Dazu stehen mehrere Möglichkeiten zur Verfügung, die Sie erkunden und bewerten sollen.

1. Aufgabe: (4 Punkte)

In dem Zusammenhang, die Übungsbeispiele des Informatik-Lehrbuch-Klassikers „Essentials of Programming Languages (Friedman, Wand, Haynes, McGraw-Hill 1992)“ nicht wie vorgesehen in Scheme, sondern in Python zu lösen, wurden von Sam Rushing für die einzelnen Übungsaufgaben Scheme-Interpreter geschrieben¹. Erkunden Sie den Interpreter für die Übungsaufgabe 9.4 und versuchen Sie, das Labyrinth-Beispiel umzusetzen.

Hinweis: Im BSCW-Bereich finden Sie das Paket und das dafür benötigte KwParsing-Modul von Aaron Watters.

- Welche Vorteile sehen Sie in der Nutzung des Interpreters in einer Unterrichtsreihe, in der Aspekte der funktionalen Programmierung thematisiert werden? (1 Punkt)
- Beschreiben Sie die Grenzen des Interpreters insbesondere im Bezug auf das Labyrinth-Beispiel. (1 Punkt)
- Beurteilen Sie den Einsatz des Interpreters in einer Unterrichtsreihe der Sekundarstufe II. Nennen Sie dabei auch positive Einsatzmöglichkeiten. (2 Punkte)

2. Aufgabe: (6 Punkte)

Für die funktionale Programmierung lassen sich auch die funktionalen Eigenschaften der Programmiersprache Python nutzen. Sie finden unter <http://www.norvig.com/python-lisp.html> eine tabellarische Übersicht von Peter Norvig und unter http://gnosis.cx/publish/tech_index_cp.html Anleitungen für die funktionale Programmierung in Python von David Mertz (Kapitel 13, 16 und 19).

- Setzen Sie mit Hilfe der o.g. Quellen das Labyrinth-Beispiel in Python um. *Hinweis:* Im BSCW-Bereich ist exemplarisch die Funktion `weg?` umgesetzt. (3 Punkte)
- Beschreiben Sie die Grenzen der funktionalen Programmierung in Python insbesondere im Bezug auf das Labyrinth-Beispiel. (1 Punkt)
- Beurteilen Sie den Einsatz der funktionalen Programmierung in Python in einer Unterrichtsreihe der Sekundarstufe II. Nennen Sie dabei auch positive Einsatzmöglichkeiten. (2 Punkte)

Abgabe: 4.7.2003

Besprechung: 10.7.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹siehe <http://www.nightmare.com/software.html#EOPL>

Aufgabe 1)

a) Durch einen Interpreter, können die Schüler ihren Code so schreiben, wie sie es schon gelernt haben, und sehen so, wie er in der jeweils anderen Sprache aussieht. Dadurch wird der Umstieg etwas leichter und sie sehen auch Gemeinsamkeiten der Sprachen. So werden die Berührungssängste gegenüber der neuen Sprache etwas abgebaut. Weiterhin kann man bei so einem Interpreter dann auch schauen, wie er arbeitet und den Code an sich ansehen, wodurch die Schüler die Regeln der Sprache schneller kennenlernen.

b) Interpreter haben im Allgemeinen immer den Nachteil, dass sie nicht alle Konstrukte richtig übersetzen, oder überhaupt Übersetzen können. Weiterhin kann es vorkommen, dass die Lösungen unverständlich, bzw. zu kompliziert sind, also nicht sauber programmiert sind. Mit Hand kann man halt eleganter Programmieren, oder auch den Code optimieren, da er in der anderen Sprache in abgewandelter Form vielleicht schneller ist.

c) Ich würde nicht zu viel Zeit auf den Interpretereinsatz verwenden, da es besser ist direkt die Sprache zu lernen. Wenn, würde ich ihn bei der Einführung einer neuen Sprache benutzen, so dass die Schüler die Angst vor dieser verlieren, indem sie durch den Interpreter ihre alten Codes, mit den neuen vergleichen können.

Aufgabe 2)

```
a) labyrinth = [['rein', 28],  
                [28, 37],  
                [37, 17],  
                [37,47],  
                [47,46],  
                [46,66],  
                [46,45],  
                [45,35],  
                [35,25],  
                [35,33],  
                [33,'raus']]
```

```
def weg(location, maze):  
    if (maze == []):  
        return []  
    elif (location == maze[0][0]):  
        return maze[0][1]  
    elif (location == maze[0][1]):
```

```

        return maze[0][0]
    else: weg(location, maze[1:]) # Peter Norvig: (cdr x) <==> x[1:] but
don't do this
def remove_location (a, maze):
    if (maze == []):
        return []
    elif (a == maze[0][0]):
        return maze[1:]
    else: remove_location(a, maze[1:])
def tour (start, ziel, maze):
    if (start == ziel):
        print 'geschafft'
    elif (weg(start, maze)==[]):
        print ' '
    else:
        print start
        print ' '
        tour(weg(start, maze), ziel, remove_location(start, maze))
        tour(start, ziel, remove_location(start, maze))

```

b) Ich sehe hier nicht so große Unterschiede. Es müssen an einigen Stellen etwas mehr Abfragen eingesetzt werden, damit einige Fehler nicht mehr auftreten, und es ist nicht mehr so kompakt und sauber programmiert, wie es unter Scheme aussieht, aber das war es acuh schon.

c) Ich denke, dass sich Python sehr gut auch dafür eignet, die funktionale Sichtweise den Schülern nahe zu bringen. Man wird es aber nicht schaffen, dann rein funktional zu bleiben, Dafür ist es dann besser eine reine Sprache zu nehmen. Als Einsatzmöglichkeiten bietet sich dann z.B.: solche Programme wie den ggt, oder die Fib-Zahlen an, die man funktional programmieren kann. Vielleicht läßt sich in Kombination mit Mathe auch ein Programm für Kurvendiskussionen in Phyton entwickeln.

Bearbeitung von Übungsblatt 10

Aufgabe 1:

- a) Um die Vorteile der Benutzung dieses Interpreters beurteilen zu können muss man etwas differenzieren.

Wenn man den Interpreter statt der richtigen Scheme Umgebung benutzt sind die Vorteile in sofern vorhanden das man:

- In der bekannten Programmierumgebung bleibt, wenn man auch einen anderen Syntax und ein anderes Paradigma benutzt.
- Man den Schüler/innen verdeutlicht, dass man auch eine Interpretierte Sprache benutzen kann, um einen Interpreter zu schreiben.

Im Gegensatz zur Benutzung von Python zur Verdeutlichung des funktionalen Paradigmas hat man den Vorteil, dass man eine vollständig funktionale Sprache benutzt, und somit das Abrutschen der Schüler/innen in ein bekannteres Paradigma effektiv verhindert.

- b) Die Möglichkeiten des Interpreters sind sehr begrenzt, da es sich um keine vollständige Scheme Umsetzung handelt. Dieses wird schon im ersten Moment des Versuches der Umsetzung des Labyrinthbeispiels deutlich. Schon die Definition des Datentyps unseres Labyrinths, mit Hilfe einer Liste, schlägt fehl. Der Interpreter unserer Meinung nach absolut nicht mit der „richtigen“ Scheme-Interpreter-Umgebung ‚Dr. Scheme‘ vergleichbar.
- c) Der Sinn des Einsatzes dieses Interpreters im Unterricht der Sek II ist trotz seiner Eindeutigen Grenzen dennoch nicht absolut Fragwürdig. Durch die oben erwähnten Vorteile kann man sagen, dass er für kleine Beispiele seinen Zweck durchaus erfüllt. Er kann z.B. durchaus als eine Art Übergang von der Programmierung in Python zu Scheme benutzt werden. Außerdem ist das Labyrinth-Problem als durchaus Komplex zu betrachten. Es wäre in Betracht zu ziehen ob man ihn bei einem, ohnehin durch den Zeitlichen Rahmen sehr begrenzten, Grundkurs als Umgebung für das funktionale Paradigma verwendet. Die Vorteile gegenüber der funktionalen Programmierung in Python sind schließlich nicht außer Acht zu lassen. Die Schüler/innen bleiben in einer gewöhnten Umgebung (Python) aber können nicht so schnell von dem geplanten Paradigma abweichen.

Aufgabe 2:

- a) Implementierung:


```

import sys
labyrinth = [['rein', 28],
             [28, 37],
             [37, 17],
             [37, 47],
             [47, 46],
             [46, 66],
             [46, 45],
             [45, 35],
             [35, 25],
             [35, 33],
             [33, 'raus']]

def weg(location, maze):
    if (maze == []):
        return []
    elif (location == maze[0][0]):
        return maze[0][1]
    elif (location == maze[0][1]):
        return maze[0][0]
    else: weg(location, maze[1:])
def removelocation (a, maze):
    if (maze == []):
        return []
    elif (a == maze[0][0]):
        return maze[1:]
    else: return removelocation(a, maze[:0])
def tour (start, ziel, maze):
    if (start == ziel):
        sys.stdout.write("%s " % start)
        sys.stdout.write("geschafft! ")
    elif (weg(start, maze)==[]):
        sys.stdout.write(" ")
    else:
        sys.stdout.write("%s " % (start))
        tour(weg(start, maze), ziel, removelocation
(start, maze))
        tour(start, ziel, removelocation(start, maze))

```

- b) Die Grenzen der funktionalen Programmierung in Python sind darin begründet, dass Python von Grund auf keine funktionale Sprache ist. Aktionen auf Datentypen sind Methoden von Objekten und keine Funktionen. Dies äußert sich bei dem Labyrinth-Problem explizit bei unserer Liste als Datentyp, und dem hierauf besonders zugreifenden Algorithmus (Funktion) ‚removelocation‘.
- c) Die funktionale Programmierung mit Python in der Sek II sehen wir durchaus als Sinnvolle Methode an. Da die Schüler/innen hier mit einer ihnen bekannten Programmiersprache arbeiten können.

Probleme sind allerdings die strikte Objektorientierung von Python, welche dafür sorgt, dass das funktionale Paradigma nicht vollständig durchgehalten werden kann.

Natürlich kann die Programmierung in Python nicht mit dem Lerneffekt einer vollständig funktionalen Sprache wie Scheme mithalten. Allerdings sind die Vor- und Nachteile die durch das Erlernen einer neuen Sprache entstehen, auf dem Kursniveau zu beurteilen.

Bei einem Leistungskurs sollte man von den Schüler/innen erwarten können, dass sie sich schnell in einer neuen Programmiersprache zurechtfinden, und somit wäre dort die Verwendung von Scheme wesentlich sinnvoller. Bei einem Grundkurs hingegen wäre es vielleicht eher sinnvoll, nur das neue Paradigma zu lehren, und ansonsten in bekannteren Gebieten zu bleiben.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 11

Sie haben in den vorangegangenen Übungsaufgaben Implementationen eines Labyrinthbeispiels in unterschiedlichen Paradigmen bearbeitet. Thema dieses Übungsblatts ist die Umsetzung in den unterrichtlichen Kontext der Sekundarstufe II auf der Grundlage des im Skript vorgestellten Modulkonzepts (Kapitel 6).

1. Aufgabe: (8 Punkte)

Formulieren Sie einen Umsetzungsvorschlag für ein Modul „Informatische Modellierung“. Überlegen Sie hierbei auch, wie Sie die „Nähe zur Mathematik“ überwinden können, indem Sie geeignete Anwendungsfälle betrachten.

- a) Geben Sie die (Lern-)Ziele an (ca. ½ Seite). (2 Punkte)
- b) Charakterisieren Sie Ihren Vorschlag im Bezug auf unterschiedliche Lernertypen, Vorkenntnisse, etc. (2 Punkte)
- c) Stellen Sie dar, welche Probleme bei der Umsetzung des Moduls auftreten können (1 Punkt)
- d) Skizzieren Sie den zeitlichen Verlauf einer Unterrichtsreihe. Welche konkreten Aufgaben würden Sie einsetzen? Wo sehen Sie Anknüpfungspunkte an weitere Unterrichtsreihen? (3 Punkte)

2. Aufgabe: (2 Punkte)

Um über den Stand der Schulinformatik argumentieren zu können, ist es unabdingbar, aktuelle Statistiken nutzen zu können. Ergänzen Sie die im Skript angegebenen Tabellen 13.1, 13.2, 13.3 und 13.5., indem Sie

- a) [MSJK2002a]¹ und [MSJK2002b]² heranziehen. (1 Punkt)
- b) versuchen, durch Recherche oder Anfragen an das MSJK weitere Daten zu erhalten. (1 Punkt)

Abgabe: 11.7.2003

Besprechung: 17.7.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹[MSJK 2002a] MSJK: Lerngruppen, Teilnehmer und erteilter Unterricht – Schuljahr 2001/02. August 2002a. – MSJK – Ministerium für Schule, Jugend und Kinder des Landes Nordrhein-Westfalen

http://www.bildungsportal.nrw.de/BP/Schule/System/Statistik/2001_02/jUnter01.pdf – geprüft: 1.7.2003

²[MSJK 2002b] MSJK: Lerngruppen, Teilnehmer und erteilter Unterricht – Schuljahr 2002/03. Dezember 2002b. – MSJK – Ministerium für Schule, Jugend und Kinder des Landes Nordrhein-Westfalen

http://www.bildungsportal.nrw.de/BP/Schule/System/Statistik/2002_03/jUnter02.pdf – geprüft: 1.7.2003

1. Aufgabe:

In den Richtlinien (S.25) wird der Informatikunterricht der Sekundarstufe II als spiralförmiger Prozess beschrieben.

In Anlehnung daran, müssen bei der Formulierung eines Umsetzungsvorschlag für die gesamte Sekundarstufe II einfache Ideen und Konzepte der informatischen Modellierung in der 11 Klasse erarbeitet werden, die dann in den Klassen 12 und 13 genutzt werden um den Stoff weiter zu durchdringen und von anderen Seiten zu beleuchten.

Unter 2.2.1.11 der Richtlinien wird erläutert, was unter Modellieren und Konstruieren im Sinne der Schul Informatik verstanden wird. Die dort genannten 4 Kriterien ließen sich ebenfalls im Rahmen des Mathematikunterrichts verwirklichen. Um eine deutliche Abgrenzung zur Mathematik zu schaffen, würde sich ein Anwendungsbereich wie „Datenschutz und Datensicherheit“ eignen, da hier keine für den Mathematikunterricht typischen Beispiele genutzt werden. Statt dessen hat man es hier mit einer Verknüpfung von Ideen aus Mathematik und Sozialwissenschaften zu tun.

Modelle für den Zusammenhang von Sender, Botschaft und Empfänger entwerfen. An diesem „einfachen“ Modellen Überlegungen zur Computer gestützten Umsetzung machen, die letztendlich zu Programmen bzw. Programmentwürfen führen sollen. Diese Programme können wahlweise Parallel (d.h. von verschiedenen Gruppen zeitgleich) oder Nacheinander in den verschiedenen Paradigmen umgesetzt werden.

Durch Überlegungen, sollen in späteren Schritten diese Modelle durch Mithörer, Schlüssel, Verschlüsselung, u.ä. erweitert werden. Diese erweiterten Modelle sollen dann wiederum am Computer umgesetzt werden. Dabei kann zunächst auch die Umsetzung in allen Paradigmen gefordert werden. Anschließender kritischen Vergleich der so gewonnenen Resultate.

a) (Lern-)Ziele:

- Erlangen von grundlegenden Kenntnissen in den Bereichen:
 - Abstrahieren aus Anwendungsbeispielen
 - Modellieren von Situationen und Zusammenhängen
 - Erkennen von Problemen
 - Modellieren von Problemen
 - Strukturieren von Lösungen

- Entwerfen von Lösungsstrategien
 - (Datenstrukturen, die für eine Umsetzung gebraucht werden)
 - Gruppen- und Team- Arbeit
 - (Algorithmen, die für eine Umsetzung gebraucht werden)
 - Imperativer Ansatz zur Problemlösung
 - Objektorientierter Ansatz zur Problemlösung
 - Wissensbasierter Ansatz zur Problemlösung
 - Funktionaler Ansatz zur Problemlösung
 - Programmiersprachen (Grundlegende Befehle, Aufbau, Möglichkeiten, Geschichte, ...)
 - Vergleich von verschiedenen Ansätzen, Lösungen oder Verfahren
 - Offenheit für neue Probleme und Fragestellungen
 - Selbständiges Arbeiten (bzw. Erarbeiten von Themen)
-
- Erlangen von Sicherheit im Umgang mit folgenden Verfahren und Werkzeugen:
 - Abstrahieren aus Anwendungsbeispielen
 - Modellieren von Situationen und Zusammenhängen
 - Erkennen von Problemen
 - Modellieren von Problemen
 - Strukturieren von Lösungen
 - Entwerfen von Lösungsstrategien
 - Gruppen- und Team- Arbeit
 - Imperativer Ansatz
 - Objektorientierter Ansatz
 - Wissensbasierter Ansatz
 - Funktionaler Ansatz

- Programmiersprachen
 - Vergleich von verschiedenen Ansätzen, Lösungen oder Verfahren
 - Selbständiges Arbeiten
- Erlangen von umfassende Kenntnisse über:
 - (Datenstrukturen)
 - (Algorithmen)
 - Funktionalität eines Rechners
 - Mind. eine Programmiersprache
 - Verschiedene Paradigmen
- Vergleichen und wählen können zwischen verschiedenen:
 - Ansätzen
 - Modellen
 - (Datenstrukturen)
 - (Algorithmen)
 - Paradigmen
 - Arbeitsweisen
 - Programmiersprachen

b)

Bei dem vorgeschlagenem Modul bieten sich verschiedenste Möglichkeiten an damit umzugehen. Abhängig von der Lerngruppe kann ein Einstieg in das Thema über einen Text (Zeitungsartikel) geschehen, welcher sozialkritisch orientiert ist oder (und) schon die Zielsetzung der Computerunterstützten Umsetzung verdeutlicht. Damit ist die Motivation für die spätere Auseinandersetzung mit Datenschutz und Datensicherheit unabhängig von den Vorkenntnissen und vom Lerntyp gegeben.

Bei der Modellierung existieren mehrere grundlegende Ansätze, abhängig vom Lerntyp und vom Vorwissen wird jeder Schüler das ihm am geeigneten erscheinende Modell nutzen. Dabei können graphische Modellierungen neben Umsetzungen auf die Datenstruktur oder formelartige Modelle

auftauchen. Wichtig ist das im folgendem Prozess alle Modelle zur Kenntnis genommen werden und auf ihre Vollständigkeit und Umsetzbarkeit am Computer überprüft werden. Fehlen wichtige Komponenten in den Modellen oder erscheinen sie nicht geeignet, so muss in anderer Gruppenkonstellation zu einem oder mehreren neuen Modellen gefunden werden.

Auch bei der späteren Umsetzung in Programmen kann durch verschiedene Modelle und verschiedene Vorgaben eine breite Palette an Möglichkeiten angeboten werden, in denen die Schüler selbständig arbeiten können. Hierdurch kann eine Leistungsdifferenzierung erfolgen, die es den Schülern ermöglicht ihr Wissen auszubauen ohne durch mangelnde Vorkenntnisse oder durch zu viel Wissen gestört zu werden. Dabei können wenig bekannte Programmiersprachen und eine Zuteilung der Schüler auf bestimmte Aufgaben hilfreich sein.

In dem immer wieder die Ergebnisse der einzelnen Gruppen zusammen getragen werden, kann die ganze Klasse am Ende der Unterrichtsreihe auf einem Wissensstand sein.

c)

- Da die Aufgaben an Anwendungsbeispielen präsentiert werden, sind Texte oder eine mündliche Einweisung in die Problemstellung notwendig. Dabei kann es sein, dass Schüler Schwierigkeiten haben das wesentliche aus den Texten zu erfassen.
- Da Kenntnisse in den Bereichen Algorithmus und Datenstruktur vorausgesetzt werden, können Schüler ohne diese Kenntnisse existieren.
- Das Modul ist auf die gesamte Sekundarstufe II bezogen, abhängig von der Schule würde das aber Lehrerwechsel nach sich ziehen. Dadurch kann es dazu kommen das es nicht in der Gesamtheit auszuführen ist, oder weitere Probleme durch nicht vorhandene aber vorausgesetzte Vorkenntnisse entstehen.
- Im Fall das verschiedene Kurse zusammengelegt werden oder Schüler in die Klasse wechseln, verfügen nicht mehr alle über die gleichen Kenntnisse.
- Es können Schüler existieren die privat oder in vorherigen Kursen in der Schule schon Kenntnisse über Informatik gesammelt haben. Daraus entsteht eine Klasse mit verschiedenem Kenntnisstand.
- Verfügen nicht alle Schüler über die gleichen Grundlagen, so können Probleme durch Unter- oder Überforderung auftreten.
- Wie in jedem Unterricht besteht die Möglichkeit, dass die Ideen der Schüler stark von denen abweichen, welche die Lehrperson im vornherein hatte. Von der Lehrperson wird hier eine hohe Flexibilität gefordert, bei der dennoch die Richtlinien nicht aus den Augen zu verlieren sind.
- Die Schüler können Schwierigkeiten haben innerhalb von Gruppen produktiv zu arbeiten.

d)

Informatische Modellierung /Block1/ Klasse 11 (Schulstunden)

- Einführung in das Thema mit Texten und geschichtlichem Hintergrund *Schüler können ihren Mitschülern den Hintergrund als Referat zu präsentieren. Gerade für Schüler mit Schwierigkeiten bietet sich damit ein guter Einstieg in eine neue Unterrichtsreihe.* (1)
- Vorstellen des ersten Beispiels zum modellieren *Objektorientiertes Modellieren und seine Darstellungsmöglichkeiten an Konkreten Beispielen einführen, anschließend die Schüler selber Aufgaben lösen lassen und die Ergebnisse vergleichen.* (2)
- Wissen über Algorithmen und Datenstrukturen aus vorherigen Unterrichtsreihen nutzen.
- Wiederholung der notwendigen Vorkenntnisse s.o. *Anhand von Aufgaben und Texten können die Schüler ihr Wissen überprüfen und wieder auffrischen.* (1)
- Entwickeln einer weiteren Methode zum Modellieren *Wiederholung des Objektorientierten Modellierens an einer Aufgabe, die später mit dem neuen Verfahren weiter bearbeitet werden soll. Anhand der Kenntnisse von Datenstrukturen soll eine Möglichkeit aufgezeigt werden ein Modell zu schaffen das ohne große Probleme auf dem Computer umgesetzt werden kann. Dabei sollen folgende Aufgabestellungen die Schüler lenken: - Wie kann eine Botschaft dargestellt werden?, Welcher Datentyp eignet sich zur Darstellung von Personen?, Erkläre warum du dich so entschieden hast. – Anschließend muss ein gemeinsames Verfahren für die Klasse gefunden und festgehalten werden. Umsetzen des Verfahrens an der oben genannten Aufgabe.* (3)
- Kenntnisse aus einzelnen Programmiersprachen nutzen, dass in anderen Unterrichtsreihen angelegt wurde.
- Kennen lernen des ersten Paradigma *Erklärungen zum Imperativen Ansatz. Den Schülern mittels texten und Beispielen die Idee dieses Ansatzes vermitteln. Die Schüler selbständig kleine Aufgaben zum Imperativen Ansatz lösen lassen. Das letzte Modell zu der übergreifenden Aufgaben in eine imperative Struktur bringen* (2)
- Imperatives Programmieren *Erklären und Aufschreiben der benötigten Befehle in einer geeigneten Programmiersprache. In zweier Gruppen (Lehrer entscheidet wer mit wem.) lösen von Aufgaben zum imperativen Programmieren. Umsetzen der übergreifenden Aufgabe am Rechner.* (3)
- Zusammenhang schaffen *In weiteren Aufgaben, ein Problem vom ersten Modell zum fertigen Programm zu bringen. Üben und wiederholen* (4)



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 12

1. Aufgabe: (6 Punkte)

Im Tagespraktikum erhalten Sie die Aufgabe, die Unterrichtsvorbereitung einer konkreten Stunde für einen Grundkurs der Jahrgangsstufe 11.1 zum Thema „Objektorientierte Modellierung am Beispiel des Schülercomputerarbeitsplatzes“ durchzuführen. Sie erhalten dazu von dem beteiligten Fachlehrer eine Planungsskizze der Unterrichtsstunde aus dem Jahr 1999¹.

Veranschaulichen Sie auf jeweils ca. 1 Seite

1. Lern- und Lehrvoraussetzungen,
2. Didaktische Gegenstandsanalyse,
3. Unterrichtsorganisation.

Nutzen Sie dazu die nach Wolfgang Klafki² und von Andreas Schwill³ entwickelten Muster. Hinweise: Da es sich im Rahmen dieser Übung um eine fiktive Schülerinnengruppe handelt und Ihnen nicht alle Einzelheiten bekannt sind, ist es notwendig, (plausible) Vermutungen als Fakten darzustellen. Sie müssen nicht alle Fragen beantworten, sondern im Wesentlichen eine plastische Darstellung der geplanten Unterrichtsstunde und ihren Kontext darstellen.

2. Aufgabe: (4 Punkte)

Von Fricke und Völter⁴ wird der Ansatz verfolgt, mit Hilfe von Entwurfsmustern die Komplexität des Planungsprozesses für die Vorbereitung, Planung, Durchführung und Reflexion von gestalteten Lernprozessen zu unterstützen. Das entwickelte Schema kann auch helfen, typische Fragen zu Problemen im Lehr-Lern-Prozess zu beantworten. Zeigen Sie mindestens vier Lösungsmöglichkeiten zu dem im Skript⁵ angegebenen Beispielproblem: „My sessions are boring, I do not feel I can engage the participants“ auf.

Abgabe: 18.7.2003

Besprechung: 24.7.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹ Sie finden diesen und weitere Unterrichtsentwürfe im BSCW-Ordner „Beispiele für Unterrichtsentwürfe“.

² Skript-Anhang A.1

³ <http://ddi.cs.uni-potsdam.de/Lehre/SPS/MerkblattUnterrichtsplanung.htm> oder Skript-Anhang A.2

⁴ [Fricke und Völter 2000] FRICKE, Astrid ; VÖLTER, Markus: SEMINARS – A Pedagogical Pattern Language about teaching seminars effectively. July 2000. <http://www.voelter.de/data/pub/tp/tp.pdf> oder <http://www.voelter.de/data/pub/tp/html/> – geprüft: 9. Juli 2003

⁵ Tabelle 16.1

Übungsblatt 12

Aufgabe 1:

1) Lern- und Lehrvoraussetzungen

Lernvoraussetzungen:

- Den Schülerinnen muss der Schülercomputerarbeitsplatz bekannt sein. Sie müssen sich mit den Grundprinzipien der objektorientierten Modellierung schon beschäftigt, und sie verstanden haben. Die Schülerinnen müssen einfache Klassenstrukturen kennen, und diese Erstellen können.
Um dieses zu sicher ist es Notwendig, dass man diese Prinzipien und Arbeitsweisen am Beginn der Stunde wiederholt, und das Verständnis durch die Schüler sichert.
- Die Schülerinnen solten Grundlegendes wissen über die Funktion des Schülercomputerarbeitsplatzes und die zugrundeliegenden Prinzipien haben, und diese in der Stunde, themenspezifisch, anwenden können.
- Durch die konkrete Beschäftigung mit den Schülercomputerarbeitsplatz haben die Schülerinnen eine weitreichende Motivation, da sie im Informatikunterricht Diesem besondere Beachtung schenken.
Die Methode der Partnerarbeit ist besonders zu berücksichtigen, wobei hierbei auf die paarung Leistungsschwacher und Leistungsstarker Schülerinnen geachtet werden muss. Die Partnerarbeit der Schülerinnen wird an eben diesen Schülercomputerarbeitsplätzen durchgeführt, zum Einem damit die Schülerinnen mit Interesse arbeiten, zum Anderen weil die Kommunikation an eben Diesen besonders hoch ist.
- Die notwendigen Kenntnisse von HTML zur Präsentation und Ergebnissicherung beherrschen die Schüler bereits.

Lehrvoraussetzungen:

- Das Interesse des Lehrenden an der objektorientierten Modellierung ist durchaus als hoch anzusehen. Er beschäftigt sich seit mehreren Jahren mit Dieser.
Das Interesse der Schüler an solchen doch sehr theoretischen Aspekten der Informatik ist erfahrungsgemäß als gering einzuordnen. D.h. Ihre Bedürfnisse nach Konkretion, im sinne von Modellierung im bezug auf Algorithmik wird nicht besonders erfüllt werden.
- Die Medialen Voraussetzungen (Schülercomputerarbeitsplatz, Beamer) sind glücklicher Weise an der Schule gegeben, und stehen für die Unterrichtsstunde uneingeschränkt zur Verfügung.

2) Didaktische Gegenstandsanalyse

- Die objektorientierte Modulierung ist eines der (Kern-)Paradigmen der Informatik und sollte somit als extrem Wichtiges Unterrichtsthema behandelt werden.
- Der Gegenstand der objektorientierten Modellierung des Accounting ist repräsentativ für alle internen Prozesse eines Informatiksystems, außerdem ist die Modellierung mit einfachen Klassenstrukturen ein wichtiger Grundsatz des objektorientierten Paradigma.
- Der Konkrete Gebrauchswert ist der, dass sie an einem konkreten Beispiel einfache Klassenstrukturen erarbeiten, die sie auch später auf Abstraktere anwenden können müssen.
- Der Sachverhalt ist nicht sehr Komplex, d.h.: Die Schülerinnen kennen die Prozesse des Accountings und können so an einem bekannten Punkt ansetzen um neue Fähigkeiten zu erlernen.
- Die Erstellung von Konkreten Klassendiagrammen ist eine der Grundfähigkeiten die die Schülerinnen beherrschen müssen um objektorientiert Arbeiten zu können. Die objektorientierte Sichtweise ist eine fundamentale Problemlösungs Struktur.

Stundenlernziel:

Die Schülerinnen erhalten einen Einblick in die objektorientierte Beschreibung des Computerarbeitsplatzes.

Feinziele:

Die Schülerinnen

- beschreiben ihren Schülercomputerarbeitsplatz mit Hilfe einer einfachen Klassenstruktur, bestehend aus Bildschirm, Tastatur, Maus.
- Entdecken, dass eine Klasse Stift nötig ist, um den Bildschirminhalt aktiv nutzen zu können.
- Beschreiben vermutete Eigenschaften (Attribute) und Funktionen (Methoden), die die Klassen Bildschirm, Tastatur, Maus und Stift für die Arbeit haben können/sollen.
- Füllen eine grafische Klassenbeschreibung mit den Attributen und Methoden.

3) Unterrichtsorganisation

Der Unterrichtsverlauf ist in acht Phasen unterteilt.

Den Einstieg bildet eine Präsentation des Schülercomputerarbeitsplatzes durch den Lehrer diese ist mit 5min zu veranschlagen, die Methode ist Lehrervortrag.

Darauf folgt für 5min eine Erarbeitung und Erleuterung und der Problemstellung in form eines mit Leitfragen versehenen Unterrichtsgespräch.

Die dritte Phase bildet die Erarbeitung der Lösung durch die Schüler hier wird die Methode der Partnerarbeit verwendet. (15mi)

Die vierte Phase ist die Zwischensicherung der Ergebnisse durch die Schülerinnen.
Die fünfte Phase ist das Unterrichtsgespräch in dem man auf die auftauchenden Probleme eingeht.

(Zeitramen 4.&5.Phase: 5min)

Die Sechste Phase bildet wieder Partnerarbeit der Schüler in der sie die Klassen ergänzen.
(10min)

Die letzten fünf Minuten sind für die Repäsentation der Ergebnisse und deren Sicherung für die Schüler vorgesehen. Außerdem wird ein Ausblick auf die Hausaufgabe gegeben.

Medien:

- Beamer, im Informatikraum vorhanden und besonderst gut zur Ergebnispräsentation geeignet
- Computerarbeitsplätze
- Lehrervortrag

Die Ergebnissicherung wird durch die Ergebnispräsentation vorgenommen. Ebenfalls sind die erstellten HTML-Dokumente hierfür gedacht.

Da die Stude die Erste einer Doppelstunde ist, wird auf Hausaufgaben verzichtet.

2. Aufgabe:

Die von Fricke und Völter entwickelten Schemata (Patterns) geben uns mehrere Lösungsmöglichkeiten für die Problemstellung „My sessions are boring, I do not feel I can engage participants“

- Eine Lösung wäre das Pattern „Change Media“
Hierbei sollte man auf folgendes Achten: Das Medium sollte passend zu der gerade bestehenden Fragestellung sein.
So ist zum Beispiel die Tafel besonders gut für dynamische Entwicklungsprozesse vor den Augen der Teilnehmenden geeignet. Eine Beamerpräsentation ist nur für Statische Inhalte geeignet. Ferner sollte beachtet werden das die Darstellungsform gut Leserlich und Optisch verständlich, ja geradezu einleuchtend ist.
- Eine weitere Lösung wäre das Pattern „Relevant Examples“
Hierbei sollte vor allem beachtet werden das die Beispiele den Teilnehmenden nicht völlig unbekannt seien dürfen. Beispiele sind dazu gedacht Probleme erklärend zu verdeutlichen, oder sogar einen Lösungsweg aufzuzeigen, und nicht etwa neue Probleme zu erschaffen.
Ideale Beispiele nehmen nicht nur Bezug auf die relevante Problemstellung, oder ihre Lösung, sondern auch auf die Teilnehmenden.
- Ein weiteres Pattern was Anwendung finden könnte wäre „Body language“
Man sollte als Vortragender (Lehrender) immer bedenken das 70% aller Information über die

Körpersprache vermittelt wird.

Man sollte seine Gestik benutzen um die Problemstellungen zu verdeutlichen. Hierbei sollte man jedoch aufpassen in kein Extrem zu geraten. Willdes Gestikulieren und hektische Bewegungen wirken eher abschreckend als Verständnisstiftend. Und eine zu passive Körperhaltung führt garantiert zu langeweile der Teilnehmenden.

- „Reference the Plan“

Auf keinen Fall sollte man die Teilnehmenden über den Sinn und zweck einer Veranstaltung im Unklaren lassen. Sie sollten ständig in der Lage sein die Gedankengänge zu verfolgen, und dabei das Ziel des ganzen Stehts voraugen haben. Das dramatische Mittel des Spannungsaufbaus durch unwissenheit funktioniert bei Lehrveranstaltungen nur äußerst selten, höchstwahrscheinlich überhaupt nicht.

Ebenso sollte man den Teilnehmenden auch nie verschweigen das es mehrere Möglichkeiten zur Lösung eines Problemes gibt. Man sollte ihnen deswegen auch darlegen warum man sich für gerade die benutzte entschieden hat.

Übungsblatt 12

1. Aufgabe

1) Lernvoraussetzungen:

- Kennen der Konzeption und Funktionsweise von Computern
- Die Schülerinnen kennen die Modellierung als idealisierte, vereinfachte Beschreibung eines realen Systems.
- Kennen von Begriffen wie:
 - Objekte als Instanz seiner Klasse
 - Klassen, Klassenbildung
 - Attributname und -Wert
 - Funktionen (Methoden)
 - Methoden

- Die Schülerinnen können HTML Dokumente erzeugen, öffnen, etc. Sie können hier eine Tabelle erstellen.

Lehrvoraussetzungen: Quellen: Wolfgang Klafki : Die bildungstheoretische Didaktik im **Sinne kritisch – Konstruktiver Didaktik**
Mathias von Saldern: „Lernen und Klassenklima“,
Ingrid Schubert

Wie ist die Qualität der Lehrer-Schüler Interaktion?

- ∞ Ist der Lehrer fürsorglich?
- ∞ Wirkt er freundlich, kontaktfreudig oder distanziert?
- ∞ Sind die Schülerinnen zufrieden mit dem Lehrer?
- ∞ Ausmaß des Autoritären Führungsstils des Lehrers: wenn die Schülerinnen den Eindruck haben, dass es ihnen wenig Spielraum zur Entfaltung eigener Ideen gewährt wird, so nimmt auch die Bereitschaft zu aktiver Beteiligung im Rahmen des Unterrichts ab.
- ∞ Wie hilft der Lehrer den schwachen und unsicheren Schülerinnen? Bevorzugt (oder benachteiligt) er manche Schüler?
Die Möglichkeit eigener Bedürfnisse artikulieren zu dürfen, und das Wissen, vom Lehrer akzeptiert zu werden, sind wichtige Voraussetzungen zur Entwicklung angemessener Selbstkonzepte.
- ∞ Verhalten sich manche Schülerinnen dem Lehrer gegenüber aggressiv?
Je geringer die Aggressionen der Schüler gegen den Lehrer ausgeprägt sind, um so positiver ist die Lernsituation in der Klasse zu bewerten.

Wie sind die Beziehungen zwischen den Schülern?

- Ausmaß der Cliquenbildung; Cliquen innerhalb der Klassen werden vorwiegend durch gemeinsame außerschulischen Aktivitäten konstituiert. In Klassen, in denen ein hohes Ausmaß an Cliquenbildung zu beobachten ist, werden häufig einzelne Schülerinnen diskriminiert.
- ∞ Sind die Schülerinnen hilfsbereit und zufrieden mit den Mitschülerinnen? Dann ist die Lernleistung der Klasse hoch.
- ∞ Sind Aggressionen der Schülerinnen untereinander zu beobachten?
- ∞ Werden Schülerinnen in einer Klasse diskriminiert?

☞ Gibt es konkurrierende Schülerinnen? Diese Schülerinnen zeichnen sich besonders durch ihr Streben nach Dominanz und Neigung zur Angabe aus.

Merkmale des Unterrichts:

- ☞ Wird Leistungsdruck geübt? Die Schülerinnen empfinden Unterricht dann bedrückend, wenn ihnen wenig Zeit für selbständiges Arbeiten bleibt, zu wenige Wiederholungen durchgeführt werden und zu viele Klassenarbeiten geschrieben werden. Umfangreiche Lehrinhalte, die in rascher Folge behandelt werden, belastet die Schülerinnen besonders. Sie neigen dann dazu, ihr Unterrichtsengagement zu verringern.
- ☞ Sind die Schülerinnen mit dem Unterricht zufrieden? Werden die Lehrinhalte verständlich vermittelt und abwechslungsreich dargeboten?
- ☞ Ohne eingewisses Maß an Ordnung und Disziplin können keine geregelte Lernprozesse stattfinden.
- ☞ Fähigkeit des Lehrers zur Vermittlung von Lerninhalten; Ist die Wahl der Unterrichtsthemen und sind die Formen des Unterrichts abwechslungsreich? Welche Medien werden eingesetzt?
- ☞ Beteiligen sich die Schülerinnen an Unterricht?
- ☞ Wirken manche Schülerinnen resigniert? Wenn die Schülerinnen den Eindruck gewinnen, dass ihre Anstrengungsbereitschaft von Lehrer nicht anerkannt wird, resignieren sie.

2) Didaktische Gegenstandsanalyse

Was soll und kann der Schüler an diesem Gegenstand lernen?

Bei der Entwicklung einer Problemlösung geht man davon aus, dass die Realität stets durch Objekte beschrieben wird, die Informationen festhalten und miteinander kommunizieren können. Der Objektbegriff steht im Mittelpunkt der Betrachtung. Ein Objekt ist durch seine Attribute und Dienste charakterisiert.

Welche allgemeine erzieherische Bedeutung kommt ihm zu?

- Schülerinnen sollen aktiv lernen.
- Schülerinnen sollen kooperativ lernen.

Wie ist der Gegenstand strukturiert?

In welchem größeren Zusammenhang steht der Sachverhalt?

Der Sachverhalt gehört zum Thema objektorientierte Modellierung, Sequenz „objektorientiert allgemein“.

Welche Lernziele sollen erreicht werden?

Stundenlernziel:

Programmierkonzepte verstehen und benutzen.

Das Konzept „Objekt“ verstehen und benutzen:

- Identifikation von Objekten, Klassenbildung.
- Attribute und Methoden beschreiben
- Objektdiagramme entwerfen

Konkrete Teillernziele:

Kognitive Zieldimension:

Den Schülercomputerarbeitsplatz mit Hilfe einer einfachen Klassenstruktur beschreiben.

Die zugehörigen Attributen beschreiben. Die Methoden beschreiben.

Grafische Klassenbeschreibung mit den Attributen und Methoden entwerfen.

Affektive Zieldimension: die Schülerinnen entdecken, dass die objektorientierte

Beschreibung des Computerarbeitsplatzes sich mit einfachen Mitteln beschreiben lässt.

3) Unterrichtsorganisation

<i>Zeit</i>	<i>Phase</i>	<i>Inhalt</i>	<i>Methode</i>	<i>Form/Medien</i>
8 ³⁵	Einstieg	Der Lehrer präsentiert die Abbildung eines Computerarbeitsplatzes	Lehrervortrag	Abbildung/ Beamer
8 ⁴⁰	Problemspezifizierung Erarbeitung	Die Schülerinnen sollen ein Modell für ihren Computerarbeitsplatz bilden. Die Klassenstruktur enthält die Klassen Bildschirm, Tastatur und Maus. Die Schülerinnen entdecken, dass eine Klasse Stift nötig ist, um zeichnen zu können.	Schülerpartnerarbeit	Arbeitsblatt
8 ⁵⁰		Die Ergebnisse werden in HTML dokumentiert	Schülerexperiment, Partnerarbeit	HTML-Seite/ Computer
8 ⁵⁵	Problem	Eine Klassenbeschreibung wird mit den Attributen und Methoden entworfen.	Partnerarbeit	Arbeitsblatt
9 ⁰⁰	Erarbeitung	Die Schülerinnen ergänzen die Klassentabelle um mögliche Attribute und Funktionen	Partnerarbeit	Computer
9 ⁰⁵	Zusammenfassung	Ausgewählte Schülerlösungen werden präsentiert	Schülervortrag	Beamer
	Ausblick/Hausaufgabe	Die Hausaufgabe zur nächsten Stunde wird gestellt		Folie

2.Aufgabe

Die Information, die wiederholt wird, auch nach einem langen Zeitabschnitt, oder die man mit anderen Informationen aus dem Langzeitgedächtnis assoziieren kann, werden in dem Gehirnteil verlagert, der für das Langzeitgedächtnis verantwortlich ist. Daher ist Wiederholung und Assoziation unverzichtbar für den Lernprozess.

In der modernen Gesellschaft wird oft vergessen, dass der Mensch das Ergebnis einer langen Evolution ist. Der biologischer Ansatz untermauert die Ansicht, dass beim Mensch heute noch die Flucht- und Angriffstendenz zu erkennen ist. Die Mißachtung des Gefühlsbereichs im Unterricht kann nachteilig auf die Lernleistungen wirken.

Es gibt verschiedene Lerntypen: visual, haptisch, akustisch, Gesprächstyp. Der Lehrer muss alle Lerntypen ansprechen. „Learning bei doing“ ist die effizienteste Lernart. Die vorteilhafte Wahl der Themen ist auch wichtig.

Das Lernen in Gruppen stellt eine Form des Lernens dar, das wiederum am Erleben der Lernenden ansetzt. Der Lehrer und die Schülerinnen befinden sich nicht in Gegenüberstellung, meist mit unterschiedlichen Interessen. Der Lehrer versucht nicht autoritär zu sein.

Lösungsmöglichkeiten:

-Lerninhalte müssen verständlich dargeboten werden. Die Wissensbasis der Lernenden muss dabei berücksichtigt werden. Die fachlichen Informationen werden nicht in einem Lehrervortrag gegeben, sondern in sehr gut verständlichen Lehrtexten. Die Schülerinnen sollen beteiligt an der Planung des Unterrichts sein.

∞ Beispiele sollen nicht kompliziert und unverständlich sein. Diese sollen den Lernenden helfen, das Thema zu verstehen.

∞ Verschiedene Medien einsetzen: Folien, overhead projektor, Beamer, Skript, Tafel,etc.

∞ Verschiedene Methoden benutzen: Gesprächsunterricht, Experimentunterricht, Lehrervortrag, schülervortrag,etc.

∞ Feedback Form.



Übung zur Vorlesung „Didaktik der Informatik I“

Blatt 13

Im Zusammenhang mit dieser Übung sollen Sie sich mit der Puzzle-Methode¹ vertraut machen. Dabei handelt es sich um eine Kombination von Gruppenarbeit und autonomen Lernen. Die Inhalte werden in mehrere, voneinander unabhängige Themenbereiche aufgeteilt, die jeweils von einer Expertengruppe bearbeitet werden. Die Expertengruppen werden nach dieser Phase aufgelöst und es werden Unterrichtsrunden gebildet, in denen jeweils genau ein Experte eines Themenbereichs beisitzt.

Sie werden in der Übung am 17.7. einer von vier Expertengruppe zugeteilt

(1. Gruppe: objektorientiert, 2. Gruppe: funktional, 3. Gruppe: Logik (wissensbasiert), 4. Gruppe: Datenfluss).

In der Übung am 31.7. werden die einzelnen Elemente der Unterrichtsreihe mit dieser Methode praktisch durchgeführt und im Anschluss reflektiert.

1. Aufgabe: (4 Punkte)

Im kollaborativen Arbeitsbereich zur Veranstaltung finden Sie die Materialien für diese Übung². Lesen Sie die Dokumente `A1-Einstieg.pdf`, `A2-imperative-Loesung.pdf` und bearbeiten Sie die für Ihre Gruppe spezifischen Aufgaben. Überprüfen Sie anschließend Ihre Lösung mit der für das Selbststudium angegebenen Lösung.

2. Aufgabe: (6 Punkte)

Bereiten Sie sich **schriftlich** auf die Unterrichtsrunde vor.

Beachten Sie die in dem Dokument `M-Mini-Didaktik.pdf` angegebenen Tipps.

Abgabe: 25.7.2003

Besprechung: 31.7.2003

Die Bearbeitungen der Aufgaben sind in elektronischer Form abzugeben.

¹siehe: <http://www.educeth.ch/didaktik/puzzle/>

²Das dieser Übung zu Grunde liegende Beispiel findet sich in den Materialien der ETH Zürich zum Informatikunterricht (vgl. [Brennwalder und Stamm 1994] BRENNWALDER, Daniel ; STAMM, Christoph ; HARTMANN, Werner (Hrsg.): Gruppenunterricht zum Thema: Paradigmen von Programmiersprachen. Zürich : ETH, September 1994. – ETH – Eidgenössische Technische Hochschule Zürich – Institut für Verhaltenswissenschaft / Departement für Informatik PDF-Dokument vom 6. November 1997 <http://educeth.ethz.ch/informatik/puzzles/paradigmen/> – geprüft: 14. Juli 2003)

1. Aufgabe

Es ist ja witzlos praktisch die Lösung aus dem Netz nochmal hin zu schreiben, also lasse ich es

2. Aufgabe

1. Überblick

In meinem Gebiet ging es um die objektorientierte Programmierung. Hier habe ich gelernt was es mit der Klassenstruktur und der Vererbung auf sich hat. Weiterhin welche Methoden welche Attribute verändern dürfen.

2. Was wissen oder können die Zuhörer nachher

Sie sollen nachher wissen, was der Sinn der objektorientierten Programmierung ist und wie sie funktioniert. Weiterhin sollen sie einfache Programme selber schreiben können

3. Unterrichtsblock

- Am Beispiel Pizza die Klassenobjekte erklären
- Daran auch die Vererbung zeigen
- Unterschied zwischen Methode und Variable klar machen
- Zugriffsrechte klären, welche Methode macht was?
- Trennung zwischen den Objekten verdeutlichen
- Kontenbeispiel in Gruppenarbeit lösen lassen, aufpassen, dass nicht einer alles macht

4. Zusammenfassung

Bei der objektorientierten Programmierung gibt es eine klare Trennung zwischen den Objekten. Objekte ändern ihre Zustände selber durch aufrufe ihrer Funktionen.

Didaktik der Informatik 1

Aufgabe 1:

Aufgabe 1.1

frisch, geschält, gehackt

Aufgabe 1.2

- 1 Pizzateig
- 2 Tomaten
- 2 Mozzarella
- Pfeffer, Paprika, Basilikum

Aufgabe 1.3

geknetet, ausgerollt

Aufgabe 2.2

Wenn der Teig mit den Zutaten belegt und die Backzeit abgelaufen ist, dann ist die Pizza gebacken.

Wenn der Teig mit den Zutaten belegt und die Backzeit noch nicht abgelaufen ist, backe den Teig eine Zeiteinheit und verringere die Backzeit um 1.

Wenn der Teig mit den Zutaten noch unbearbeitet ist backe noch nicht und gebe den unbearbeiteten Teig zurück.

Aufgabe 2.3

```
data TomatenZustand == frisch++geschält++gehackt;
type Tomate == TomatenZustand;
dec TomatenSchälen:list(Tomate) => list(Tomate);
--- TomatenSchälen(frisch::rest) <= geschält::TomatenSchälen(rest);
```

Aufgabe 2.4a

```
dec TeigAusrollen:PizzaTeig -> PizzaTeig;
--- TeigAusrollen(5,4.5,geknetet) <= TeigAusrollen(7,2.25,geknetet);
--- TeigAusrollen(7,2.25,geknetet) <= TeigAusrollen(9,1.125,geknetet);
usw.
```

Die Praktikabilität und vor allem Speichereffizienz dieses Ansatzes sinkt mit der Größe der eingegebenen Variablen, die allerdings im konkreten Beispiel konstant festgelegt sind.

Aufgabe 2.4b

Theoretisch würde dieser Abschnitt nicht terminieren, bzw. eine Lösung zufällig gewählt. Bei konkreten Umsetzungen des funktionalen Paradigmas ist allerdings mit einer sequenziellen Abarbeitung der Möglichkeiten zu rechnen, sprich in diesem Fall würden Tomaten gewaschen werden.

Aufgabe 3.1

```
dec listen_summierung: list(num) -> num;
--- listen_summierung ([]) <= 0;
--- listen_summierung (erstesElement::rest) <= erstesElement + listsum(rest);
```

Aufgabe 3.2

```
dec f: list(num)#num -> list(num);
--- f([],x) <= [];
--- f(erstesElement::rest,x) <=
if erstesElement = x then
f(rest,x)
else
erstesElement::f(rest,x);
```

Aufgabe 2:

Überblick in drei Sätzen:

Die Aufgabenstellung ist mit Hilfe des funktionalen Paradigmas elegant lösbar. Allerdings ist es von sich aus keine komplette Lösung, d.h. die Lösung muss auch funktional angewendet werden, um zu richtigen Ergebnissen zu kommen. Außerdem haben wir das funktionale Paradigma noch mehr verinnerlicht.

Was wissen oder können die Zuhörer nacher:

Sie sollten die Lösung verstehen und (idealistisch) selbst verinnerlichen. Sie sollen den funktionalen Ansatz und seine Anwendung, und dessen Problematik verstehen. Sie sollen verstehen, dass funktional programmierte Teilaufgaben (Funktionen) auch funktionaler Anwendung bedürfen, um das richtige Ergebnis zu produzieren (Reihenfolge).

Unterrichtsblock/ Vortrag:

Erleuterung des funktionalen Ansatzes an der Lösung der Aufgaben mit Einblick auf die gestellten Probleme.

Speichereffizienz 2.4a

Nicht- Termination des Programms bei nicht sachgemäßer ausführung

Funktionale Ansätze allgemein:

Die funktionelles Denken als Gegensatz zu objektorientiertem oder prozeduralem Denken. Vergleiche Ziehen (Methoden von Objekten können auch Funktionen sein)

Reihenfolge:

1. Einführung in das funktionale Paradigma (grober Überblick)
2. Erläuterung der einzelnen Aufgaben an den Musterlösungen
3. Eingehen auf die sich gestellt habenden Probleme
4. Vergleiche zwischen den Paradigmen ziehen und erläutern (Verständniss-stiftend)
5. Zeit für Rückfragen der Zuhörenden