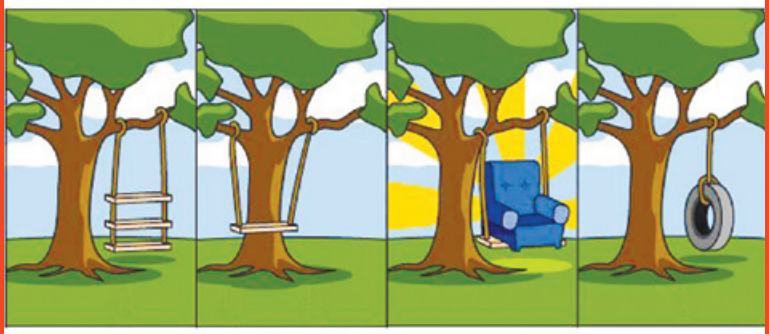
Visionen
Praktisches Lehrbuch

Software Engineering



Ein Standardwerk in sechs Bänden
Band 5
Jahrgang 2004

Ausgabe 05/2004

Magazin des Vereins der Informatik Studierenden an der ETH Zürich (VIS)

Erscheinungsweise: 6x jährlich
Auflage: 1500
Jahresabonnement: SFr. 25.-
Redaktion, Konzept & Realisation:
Alex de Spindler, Jonas Wäfler
Verlag: Beat Schwarzentrub

Mitarbeiter an dieser Ausgabe

Prof. Hans Hinterberger, Prof. Bertrand Meyer, Claude Knaus, Benno Baumgartner, Ádám Darvas, Peter Müller, Werner Dietl, Beat Schwarzentrub, Till Kleisli, Res, Melanie Rämi, Manuel Graber, Mathias Payer, Michael Grossniklaus, Jonas Wäfler

Anschrift, Verlag & Redaktion

Verein der Informatik Studierenden (VIS)
ETH Zentrum, RZ F17.1
CH-8092 Zürich
Tel.: 01 / 632 72 12
Fax: 01 / 632 16 20

Präsenzzeiten: Mo. bis Fr. 12:15 bis 13:00
Postkonto: 80-32779-3

http://www.visionen.ethz.ch/
Email Redaktion: **visionen@vis.ethz.ch**
Email Verlag: **verlag@vis.ethz.ch**

Inserate

1/1 Seite, schwarz/weiss	SFr.	750.–
1/1 Seite, s/w + 1 Farbe	SFr.	1000.–
1/1 Seite, 4-farbig	SFr.	1500.–

Andere Formate auf Anfrage.

Druck

Binkert Druck AG
Baslerstrasse 15
5080 Laufenburg
062 869 79 79

Editorial

ALEX - HOBBY METEOROLOGE

Liebe Visionenleser

Der Sommer liess klimamässig einiges zu wünschen übrig, aber das Klima ist ja nicht wirklich der Grund, weshalb wir uns in Zürich befinden, ja? Hast du dich schon mal gefragt, wieso unsere Vorgänger bei der letzten Völkerwanderung an der sie teilnahmen ausgerechnet in diesen Längen- und Breitegraden stehen geblieben sind? Waren sie zu faul, um weiter zu gehen, oder war ihnen das Wetter einfach egal? Ich habe mich diesen Sommer mit dieser Frage beschäftigt und bin dafür ein bisschen herumgereist. Die Quintessenz: Erstens, wenn du mit dem diesjährigem Sommer in Zürich nicht zufrieden warst, bedanke dich bei deinen Eltern, dass sie dich nicht im Ruhrgebiet auf die Welt gesetzt haben (falls sie den Dank nicht annehmen, rekursiv durch den Familienbaum durchgehen). Zweitens, je angenehmer das Klima, desto lockerer die Arbeitsmoral. Wir leben folglich nicht in Zürich, weil es hier herausragend schön ist, sondern weil wir fleissig sind. Jawohl und das war schon immer so. Deshalb beginnt das Studium jeweils im Herbst, weil diese Jahreszeit zu noch mehr Fleiss verhilft! Der VIS wünscht dir einen guten Semesterbeginn und hofft, mit dieser Visionenausgabe, dein Interesse für Software Engineering wecken zu können.

Titelbild:
http://wiki.ael.be/index.php/FightingSWPatentsCartoon

Zitat:
Alte Bauernweisheit.

Inhalt

Vom Departement

Herzlich willkommen!

PROF. HANS HINTERBERGER - STUDIENDELEGIERTER

Sie haben sich entschlossen, während den nächsten drei bis vier Jahren, bei uns und mit uns, die für ein ETH-Bachelordiplom in Informatik verlangten Fähigkeiten zu erwerben und haben somit den ersten Schritt in ein faszinierendes Fachgebiet getan. Sie wählten einen forschungsnahen Studiengang, welcher Ihnen diejenigen Grundlagen vermittelt, die Sie als hoch qualifizierte Informatik-Ingenieurinnen und -Ingenieure auszeichnen werden. Das dabei erlernte Wissen und Können ist mehr denn je sehr gefragt – trotz allen konjunkturellen Schwankungen.

Während Ihres Studiums werden Sie nicht nur eine Fülle an technischem Wissen aufnehmen, Sie werden auch eine neue Sprache lernen, die Sprache der technischen Wissenschaften im Allgemeinen und die Sprache der Informatik im Speziellen. Damit meine ich nicht eine Programmiersprache, sondern den Wortschatz der Begriffe und die sprachlichen Konventionen, die Sie beherrschen müssen um die dem Ingenieurwesen eigenen, komplexen Sachverhalte erfassen zu können. Die dazu notwendige Breite propädeutischer Fächer widerspiegelt sich im Lehrplan der ersten zwei Studienjahre. Im ersten Jahr beschäftigen Sie sich vorwiegend mit allgemeinen, ingenieurwissenschaftlichen Grundlagen. Im zweiten Jahr steht das Fachwissen, mit dem jede Informatikingenieurin und jeder Informatikingenieur vertraut sein muss, im Zentrum. Im dritten Jahr werden sie Ihre Kenntnisse in Spezialgebieten Ihrer Wahl vertiefen.

Das Departement Informatik betrachtet das Bachelordiplom in erster Linie als notwendige Voraussetzung für ein anschliessendes Masterstudium. Im Gegensatz zum Bachelorstudium ermöglicht Ihnen der Masterstudiengang von Anfang an die Spezialisierung in einem der zahlreichen Gebiete der Informatik. Vielleicht entscheiden Sie sich sogar für ein Masterdiplom in einem anderen Fachgebiet. Auch dieser Weg steht Ihnen offen. In jedem Fall ist es von Vorteil, wenn Sie sich bereits im zweiten Bachelorjahr überlegen werden, in welche Richtung Sie Ihre akademische Laufbahn lenken wollen. Ein Masterdiplom zeichnet Sie nicht nur als fähige Ingenieurinnen und Ingenieure aus, es gibt Ihnen auch die Grundlage für ein Doktoratsstudium.

Unabhängig davon, welche berufliche Laufbahn Sie einmal einschlagen werden gilt, dass für das erfolgreiche Lösen vieler interessanter Probleme und Aufgaben interdisziplinäre Fachkenntnisse zu einer Notwendigkeit geworden sind. Kompetenzen in mehr als einem Fachgebiet führen nicht nur zu interessanteren Aufgabenstellungen, sie können auch für den beruflichen Erfolg ein ausschlaggebender Vorteil sein. Die zweistufigen Studiengänge der ETH Zürich geben Ihnen die dazu notwendige Flexibilität.

Informatiker müssen nicht nur den Jargon der Ingenieurwissenschaften und den ihres Fachs beherrschen, sie lernen auch, mit künstlichen und formalen Sprachen umzugehen. Hier ist ebenfalls die Mehrsprachigkeit von grossem Vorteil, denn nicht alle Anwendungsgebiete bevorzugen die gleiche Programmiersprache. Andererseits muss sich das Departement aus didaktischen Gründen im Grundstudium für eine einzelne Unterrichtssprache entscheiden. Als Auswahlkriterium wird dabei nicht die Popularität einer Programmiersprache berücksichtigt, sondern die Professoren und deren Arbeitsgruppen, welche an der Entwicklung der verwendeten Sprache massgeblich beteiligt waren, noch immer sind und sich engagiert für unseren Unterricht einsetzen. Dies gewährleistet, dass Sie mit Unterrichtspersonen arbeiten werden, welche Sie kompetent unterstützen und die, soweit angebracht, ein forschungsnahes Lehren praktizieren.

Im Namen des Departements bedanke ich mich für das uns entgegengebrachte Vertrauen und versichere Ihnen, dass alle Dozierenden ihr Bestes geben werden, um Ihnen eine exzellente Ausbildung anzubieten. Das Lernen selbst können wir Ihnen allerdings nicht abnehmen. Ich bin aber überzeugt, dass Sie zielstrebig und selbständig Ihr Studium angehen und auch erfolgreich abschliessen werden und wünsche Ihnen dazu nur das Beste und viel Freude.

VisAktiv

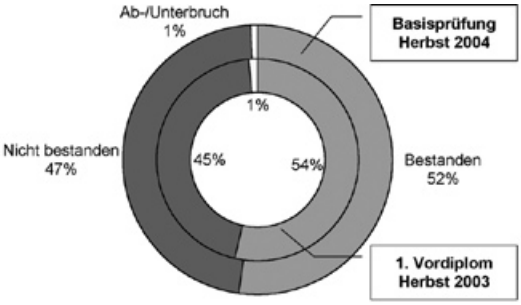
Prüfungsstatistik Herbst 2004

BEAT - CHEF-STATISTIKER

Basisprüfung

In dieser Prüfungssession traten das erste Mal Studierende im neuen Bologna-Studiensystem zur Prüfung an. Das 1. Vordiplom ist tot, es lebe die Basisprüfung!

Angemeldet für die Basisprüfung haben sich 240 Personen, ganze 10 haben die Zulassung infolge fehlender Testate aber nicht erhalten. Abzüglich ordentlicher Abmeldungen und Prüfungsabbrüche absolvierten 208 Kandidaten die Basisprüfung. Hinreichend richtige Antworten aufs Blatt gekritzelt haben davon 112 Personen. Die Durchfallquote liegt mit 44% in etwa im Rahmen der vergangenen Jahre. Man sieht also, die Basisprüfung ist eigentlich nur alter Wein in neuen Schläuchen.



Basis Prüfung

Fach	μ	s^2
Einf. Programmierung	4.22	1.14
Datenstrukturen u. Algo.	3.83	0.97
Logik	4.13	0.90
Diskrete Mathematik	3.74	0.93
Lineare Algebra	3.60	1.19
Analysis I+II	4.07	1.11
W'keit & Statistik	3.96	1.01
Physik	4.44	0.95
Digitaltechnik	3.47	1.33
Total	**3.97**	

Insgesamt **213** KandidatInnen

Bestanden	**111 (52.1%)**
Nicht bestanden	**100 (46.9%)**
Abbruch	**2 (0.9%)**

Bezogen auf **47** RepetentInnen

Bestanden	**21 (44.7%)**
Nicht bestanden	**26 (55.3%)**
Abbruch	**0 (0.0%)**

Werfen wir noch einen Blick auf die einzelnen Prüfungsfächer. Der Gesamtnotendurchschnitt sowie die Notendurchschnitte der meisten Fächer lagen im Rahmen der letzten Jahre. Auffällig von

3.83 auf 3.67 verschlechtert haben sich Diskrete Mathematik und Lineare Algebra (früher Algebra I + II). Dies mag an der teilweisen Neukonzipierung liegen, die diese Fächer erfahren haben (Verschiebung gewisser Teile aus dem zweiten Studienjahr ins erste). Analysis hat sich dafür um mehr als 0.2 Noten verbessert und ist jetzt sogar genügend. Auch in Physik gab es eine Steigerung. Dieses Fach dauert jetzt allerdings nur noch ein Semester, möglicherweise beruht das bessere Resultat darauf, dass man sich in dieser kurzen Zeit nicht mehr in die schwierigen Themengebiete vorarbeiten kann.

2. Vordiplom

Das gute alte 2. Vordiplom gehört schon bald der Vergangenheit an. Der letzte Jahrgang von Studierenden, der nach dem alten Diplomreglement studiert, hatte die Ehre, das letzte Mal ein grosses Prüfungspaket im Herbst zu absolvieren. Im Bachelor gilt dann ein Kreditsystem und die Fächer werden einzeln geprüft.

190 Personen haben sich für das 2. Vordiplom angemeldet, tatsächlich absolviert haben es schliesslich 161 Kandidatinnen und Kandidaten. Wie auch beim 1. Vordiplom sind das weniger als noch vor einem Jahr. Diese Abnahme der Studentenzahlen liegt im allgemeinen Abwärtstrend, der aber, wie schon seit längerer Zeit erfolglos behauptet wird, demnächst gestoppt wird.

Der Gesamtnotendurchschnitt ist mit 4.21 leicht besser als letztes Jahr (4.13). Bei den Fächern gab es relativ grosse Unterschiede zum Herbst 2003. Informatik III + IV verbesserten sich um 0.3 Noten, was damit zusammenhängen dürfte, dass die Vorlesung neu von einem anderen Professor mit einem anderen Konzept gehalten wurde. Wie man hört, war die Motivation der Studierenden bedeutend höher, was sich in den Noten deutlich widerspiegelte. Auch bei Elektrotechnik (+0.43) und Information und Kommunikation (+0.37) war das Ergebnis besser, wobei ich dafür aber keine wissenschaftliche Erklärung habe. :-)

Die Kandidatinnen und Kandidaten waren aber natürlich nicht durchwegs Wunderkinder. NSR/WiRe musste 0.12 Noten abgeben und rutschte noch tiefer in die Wüste der ungenügenden Noten. Auch Theoretische Informatik und Informationssysteme-G sackten ab, blieben allerdings noch knapp genügend (von 4.49 auf 4.05 bzw. von 4.43 auf 4.06). Bei TI liegt das möglicherweise auch am neuen Professor und der neuen Konzipierung, worauf ich mich aber nicht behaften lassen möchte. Vielleicht waren die Studis auch einfach schlecht :-) Ich könnte nun auch noch eine Analyse machen, ob diese Abweichungen

2. Vordiplom

Fach	μ	s^2
Informatik III + IV	4.57	0.76
NSR/WiRe	3.59	0.87
ElTech/DigiTech	4.40	0.98
SysProg	4.45	0.89
InfKomm	4.43	0.83
Vernetzte Systeme	4.29	0.86
Theoretische Informatik	4.05	0.87
Informationssysteme	4.06	0.92
Total	**4.21**	

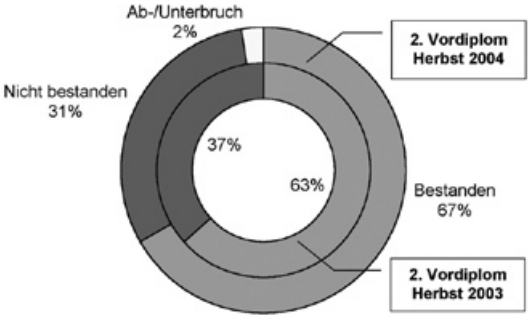
Insgesamt **165** Kandidatinnen

Bestanden	**110 (66.7%)**
Nicht bestanden	**51 (30.9%)**
Abbruch	**4 (2.4%)**

Bezogen auf **27** RepetentInnen

Bestanden	**16 (59.3%)**
Nicht bestanden	**10 (37.0%)**
Abbruch	**1 (3.7%)**

statistisch überhaupt signifikant sind, aber leider, leider habe ich hier keinen Platz mehr...

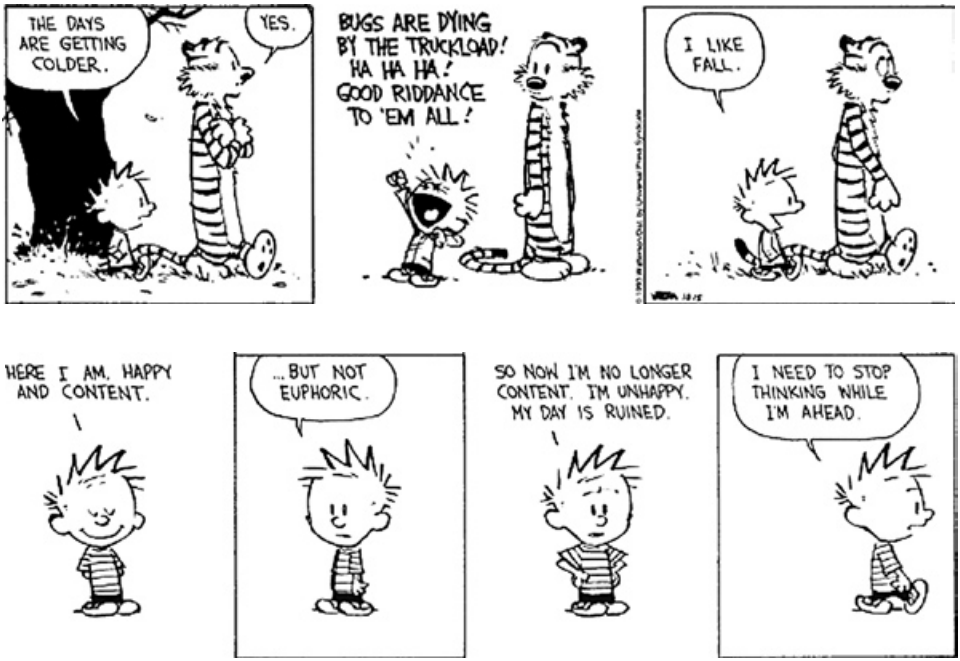


Kernfächer

174 bzw. 152 wackere Ritterinnen und Ritter wagten sich in die Höhle des Löwen und versuchten sich an Digitaltechnik und Rechnerstruktur bzw. Theoretischer Informatik. Das klappte soweit ganz gut, nur für 12 bzw. 15 Kandidaten war die Mission erfolglos. Sie müssen (oder dürfen, haha!) die nicht bestandenen Kernfächer wiederholen oder auf ein anderes ausweichen.

Diplomabschlüsse

65 Kandidatinnen und Kandidaten haben ihr Fachstudium erfolgreich abgeschlossen. 20 von ihnen dürfen sich neu mit dem Titel *Master of Science ETH in Computer Science* schmücken, die anderen sind ab sofort *Dipl. Informatik Ingenieur(in) ETH*. Wir gratulieren allen Diplomierten herzlich!



StudentAktiv

Willkommen

MELANIE - WAR AUCH MAL ERSTSEMESTRIGE

Liebe Erstsemestrige, ich möchte euch nicht nur herzlich begrüssen an der ETH, sondern euch auch noch einiges mit auf den Weg geben.

Es gibt verschiedene Motivationen mit einem ETH-Studium anzufangen: Einige von euch sind vor allem wegen des Fachwissens hier. Dies ist sehr ideal, denn ihr werdet nie mehr die Chance erhalten, so effizient so viel Wissen zu erlangen. Dazu kommt, dass alle späteren Weiterbildungen viel teurer sind. Ebenfalls loben möchte ich das breite Angebot im Fachstudium, woran sich ein wissenshungriger Student laben kann.

Bist du auch hier, um möglichst viel Wissen zu erlangen? Dann greif zu!

Dies ist mit Absicht als Aufforderung formuliert. Denn zugreifen verlangt etwas Aktives von dir. Du musst dem Wissen nicht mal nachrennen. Es wird dir erzählt, Unterlagen werden verteilt oder empfohlen, Übungsgelegenheiten sind reichlich vorhanden wie auch Gelegenheiten, um Fragen zu stellen. Es liegt alles da, vor deiner Nase, nur nehmen musst du es dir selber. Zuhören, lesen, üben, fragen und vor allem, es wollen! Aber du willst es ja, darum bist du hier. Bitte vergiss das nicht.

Eine andere Motivation für ein ETH-Studium, ist seine Grenzen zu finden. Die meisten von euch waren sehr gut im Gymnasium. Hast du dich auch manchmal gefragt, wie gut du wirklich bist? Wozu du fähig wärst, wenn die Anforderungen höher wären, die Konkurrenz härter? Wie weit du gehen könntest, wenn du gefordert und gefördert werden würdest? Dann gib alles!

Hier wirst du gefordert und gefördert soweit du mitmachst. Es gibt auch genügend Möglichkeiten, sich zu messen, und genügend Konkurrenz, die sich ebenfalls beweisen möchte. Es ist das ideale Umfeld, um seine Möglichkeiten und Grenzen zu erforschen.

Alles geben heisst, auch wirklich an seine Grenzen zu gehen und nur mit sich zufrieden zu sein, wenn man sein Bestes gegeben hat. Sein Bestes zu geben ist sehr befriedigend, und zu wissen, wo seine Grenzen sind, ist sehr beruhigend, weil man sich dann besser kennt. Ich nehme nicht an, dass viele hier sind, um später das grosse Geld zu machen. Erstens ist es fraglich, ob man heutzutage überhaupt das grosse Geld noch macht, und zweitens gäbe es dazu sicher einfachere Wege.

Unabhängig davon, welches deine Motivation ist: es ist wichtig, sie zu kennen und sie nicht aus den Augen zu verlieren. Der Vorteil ist dann nicht nur, dass du mehr weisst oder bessere Noten hast. Der Hauptvorteil ist, dass man das Studium wirklich schätzt!

Als letztes noch ein Appell an die Studenten in den höheren Semestern:

Falls ihr euren Beweggrund nicht mehr klar vor Augen habt, versucht euch daran zu errinnern, was eure Motivation war, hierher zu kommen.

Software Engineering

About the Chair of Software Engineering

BERTRAND MEYER - MONSIEUR EIFFEL

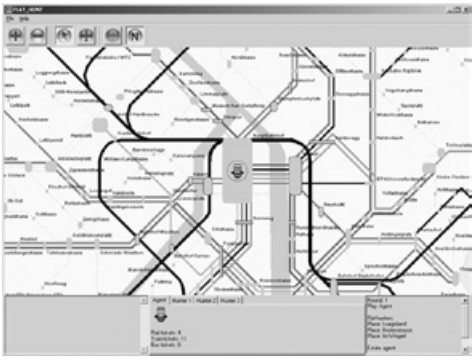
Software doesn't just happen. Behind the systems that figure ever more prominently in our daily life, from cell phones — according to Siemens, 70% of the features that determine a brand's competitive advantage are now provided by software — and cars (some new models include 10 million lines of code across all embedded processors) to the reservation systems that enable us to travel and the financial systems through which we pay and get paid, there are programs. Behind these programs there's analysis, design, implementation, testing, maintenance, documentation, configuration management, project management. And behind all of that there are people conceiving, producing and enhancing the programs: software engineers. Our task in the Chair of Software Engineering is to train these professionals and advance the state of the art in the field.

As an academic discipline, software engineering covers all the elements that help produce quality software: methods, tools, languages, processes. The focus on quality is what distinguishes this discipline from the mere task of programming; it's there to ensure that the systems we release are correct (do the job), robust (cope with abnormal situations), secure, efficient in their use of hardware resources, easy to learn, easy to use, easy to change when some new requirements pop up, and delivered on time at reasonable cost. Software engineering is also about *size*: many techniques that may be acceptable for small programs don't scale up to today's large systems — large not only in the final volume of code (hundreds of thousands or millions of lines) but also in years of development, number of changes, number of users, number of developers involved.

The techniques of software engineering, developed over the past four decades, have already had a major effect on the practice of software construction, even when practiced by people who have never taken a software engineering course: better programming languages (which enable developers to concentrate on the application area rather than low-level details, and avoid costly run-time mistakes), better development tools such as the "Integrated Development Environments" (IDEs) that have come into common use, configuration management, safer operating systems, fancy user interfaces are only some of the benefits that we all enjoy. But this is still a young discipline and many open problems remain, in both theory and applications.

There's more than one *Kultur* of software engineering. On the applied side we find work on programming languages, compilers and operating systems, which has traditionally been strong at ETH. There's the whole area, where ETH has also been prominent, of *methodology*: developing principles and techniques for analysis, design and the other

phases of software construction, emphasizing quality throughout. More on the side of experimental science we see the subcultures of *metrics*, applying quantitative techniques to analyze properties of programs and of the programming process, and of *models*, to predict software properties, for example to estimate the time and cost it will take to complete a project, or the number of bugs that remain in a delivered product. Regarding bugs indeed, or rather their avoidance, there's the whole area of quality assurance, in particular *testing*. In situations that require absolute guarantees of correctness, testing an already developed program is not enough: *formal methods* apply a strict development process based on mathematical specifications, with proofs of correctness at every step. Cutting across all these areas is the need for software engineering *tools*, from complete IDEs to tools supporting specific software tasks, for example configuration management tools that help manage the evolution of the many components of a system and avoid the sometimes catastrophic mistake of combining a version of a component with an incompatible version of another component. All these subcultures are focused on the technology; another part of software engineering concerns itself with *management* aspects: how to drive large, multi-person projects to success, how to divide the tasks, how to organize the software development process, how to keep costs under control.



FLAT_HUNT - Lernsoftware für die Erstsemestrigen

This is a wide range of topics. While in our teaching we have to touch on just about everything, our research focuses on selected key areas in which we can make a difference. While not neglecting theory, we keep a strong practical focus, with the aim of producing results that are of direct use to the software industry in Switzerland and elsewhere. Here are some of our research projects; in each of them, we offer semester and master's projects.

Much of the progress in software engineering methods has followed from *component-based development*, the idea of building software from packaged software modules, the equivalent of "chips" for electrical engineers. A central theme of our work is the notion of *trusted component*: making sure that these reusable modules are of guaranteed quality, so that we can trust the quality of the systems made by assembling many of them. This notion ranges across many issues of software engineering. For example we have developed a "code crawler", the Google of components if you like, which analyzes reusable components to explore which components they themselves rely on directly or indirectly, in the end identifying the most successful components. More generally we are developing techniques for certifying commercial components (which could be .NET assemblies, object-oriented libraries in Eiffel or other languages, Enterprise Java Beans) and a general model for defining and assessing component quality.

Object technology and object-oriented languages have revolutionized the way most of the industry programs. We have extensive background in the development of the Eiffel object-oriented method, language and tools; we continue work in this direction, refining the language and extending its capabilities. An international standard for Eiffel is in the final stages of development through ECMA, the computer standards organization; it should be com-

pleted in 2004 and, we hope, adopted in 2005, first by ECMA and then by the International Standards Organization. While standards efforts are often conservative in nature, solidifying existing practice rather than innovating, this one has been different, and includes novel programming language concepts, for example a technique for guaranteeing that «void calls» (applying an operation to a non-existent object because a pointer is null — the plague of many object-oriented programs) will never occur.

Eiffel embodies the methodological principle of «*Design by Contract*», which has influenced many other developments; the idea is to associate elements of formal specification with every unit of a program (class or routine). Design by Contract has a profound effect on the software development process; one of its benefits is to make the testing effort much more focused and effective. Based on this idea, we have started an extensive research project to make component testing completely automatic, including test generation, based on the components' contracts. Such «*Push-Button Testing*» removes the tedious manual process of preparing «test suites» and «test oracles»; all this becomes automatic, and we can let a machine run night and day by itself, looking for bugs. We have indeed already found some significant bugs in existing libraries, and envision many more developments, in particular for testing of systems with Graphical User Interfaces. This area is rich with possibilities of student projects.

Another almost endless source of projects is *concurrency*. While advances in programming languages and techniques have allowed most developers to program significantly better than 10 or 20 years ago, at a much higher level of abstraction, this progress has only affected «sequential» programming, where you are dealing with a single thread of control. With the advent of the Internet and the growing popularity of «multithreaded» environments which allow your program to do (apparently) several things at a time, more and more programmers are dabbling into concurrent programming schemes, but there the level of abstraction remains desperately low, using techniques (locks, semaphores, manual synchronization) devised in the late sixties. Reasoning about programs remains operational and low-level, and debugging of concurrent or multithreaded programs (for which bugs are usually non-deterministic and hence hard to reproduce) remains a nightmare. Our SCOOP project — for «Simple Concurrent Object-Oriented Programming» — builds on object-oriented concepts to provide a clear and safe style of programming concurrent and networking applications. This is an ambitious project: the easier it is for programmers in the end to produce efficient and reliable concurrent applications, the harder for us to build the framework that hides the underlying complexity of concurrent architectures and rules out tricky mistakes. It is starting to produce results usable in practice, across a wide range of concurrent setups, from multithreading to real-time, embedded devices and distributed systems. Although the ideas are completely general and platform-independent, we have built the current implementation on top of a specific framework, Microsoft's .NET, which provides a sound basis for both multithreading and Web service applications. Other implementations, for example in the Linux world, are planned. There have already been a number of interesting student theses in this area, and many more are possible as the technology base builds up.

We are also active in formal methods and proofs. Jean-Raymond Abrial's and Peter Müller's groups have their own attractive projects in this area; for our part we are exploring the systematic construction of object-oriented components proved correct. Imagine building your programs on the basis of reusable components equipped not just with full specifications but also with mathematical *proofs*, checked by computer, that the components' implementations

CREDIT SUISSE

Eine Karriere braucht eine Vision.
Und die Wahl des richtigen Partners.

Wir setzen auf Nachwuchstalente, die anspruchsvolle Aufgaben mit viel Enthusiasmus und Engagement angehen und ihre Karriere durch ein hohes Mass an Selbstverantwortung vorantreiben möchten. Mit einem überdurchschnittlichen Studienabschluss, Ihrer überzeugenden Persönlichkeit und ausgeprägten sozialen Kompetenzen bringen Sie die besten Voraussetzungen für Ihre Karriere bei uns mit. Attraktive Career Start Opportunities bei der Credit Suisse, der Credit Suisse First Boston und der Credit Suisse Asset Management erwarten Sie. Sind wir Partner?

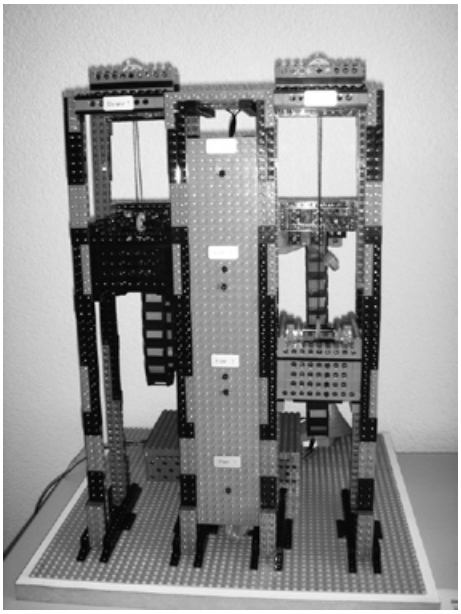
www.credit-suisse.com/careerstart

Wir suchen Hochschulabsolventen, die noch nie mit der erstbesten Lösung zufrieden waren. www.mckinsey.ch

McKinsey&Company

Ruf Lanz

actually satisfy these properties. Progress in this direction implies taking the concepts of Design by Contract further, to cover complete specifications of systems. A particular challenge is to cover a realistic object-oriented scheme where the run-time structure involves extensive use of *pointers* (or «references»), which have traditionally eluded simple mathematical specification. We have the basis of a theory in this area and have started to develop, specify and prove a realistic library of fundamental structures and algorithms. We are also looking into how to make formal development and proof systems more easy to use by ordinary developers in industry, through the development of modern tools and interfaces. If you like software work with a sound mathematical basis, in an area (general library classes) where you can have a real impact on practical software development, these topics should attract your interest.



Liftmodell zum Testen von SCOOP

The combination of formal work focused on proofs with innovation in testing is not that common; many groups are focused on one or the other approach. We feel that there is no contradiction, and that the problems of building correct software are hard enough to preclude ignoring any approach that can help. Another such unnecessary split in the software engineering community is between *components* and *design patterns*. These are two of the most promising developments in the field; we have worked to reconcile them by showing that (against the view of the original developers of patterns) a majority of patterns can be turned into reusable components, usable off-the-shelf rather than reprogrammed by every new application that needs them. This *componentization* effort has to be taken further and extended with more formal specifications.

One of the open challenges in object technology is to combine object-oriented programming with databases and other *persistence* mechanisms. While much of the industry seems to have given up on object-oriented databases, there remains a huge gap between the models we use for describing our programs' objects and those under which we store them. Work on O-O persistence decreases that gap; ideally, programmers should be able to think only of their programs' objects, letting an automatic mechanism store them away for future use and handle queries, transactions and other benefits of database technology. This area of research is attractive in particular if your interests span the worlds of software engineering and databases.

Some of our research is concerned with computer science education. In teaching the new «Introduction to Programming» course, we have developed innovative «Outside-In» techniques based on the concept of *Inverted Curriculum*; the approach is supported by a textbook in progress, «*Touch of Class*» (see [1]), and a large software system, TRAFFIC,

with multimedia and animation facilities. TRAFFIC is an almost endless source of new developments; if you fancy advanced graphics programming, have a creative mind, and like the idea of your work helping future students learn programming faster and better, consider joining TRAFFIC development.

We have many more research ideas and projects, as well as applications (for example providing a better approach to email). Take a look at [2], which has pages on the various projects mentioned above, or talk to us if you don't find what you are looking for.

Our group is also active in teaching: in the required part of the curriculum — the first two years — in addition to Introduction to Programming as mentioned, we are responsible for «Software Architecture», which I'll teach in its new form for the first time in the coming year. Jean-Raymond Abrial and David Basin will be teaching «Formal methods and functional programming». We also offer several advanced courses, from «Trusted Components: Contracts, Patterns and Reuse» in the Fall, «Object-Oriented Software Construction» in the Spring (which I would particularly recommend if you want to get an in-depth look into object technology and understand what the fuss is about), «Concurrent Object-Oriented Programming», and a new course on the fascinating topic of software outsourcing and offshoring, a phenomenon that is affecting the whole industry. Peter Müller teaches courses on programming language semantics and object technology, and Jean-Raymond Abrial on formal development.

The FATS seminar (Formal Approaches To Software) takes place on Wednesdays at 16:15 during the semesters ([3]). Our weekly research meeting is every Wednesday from 14:05 to 16, currently in IFW E42; we welcome visitors.

Our Chair is quite active on the international scene. We publish the main international journal on object technology: JOT, the Journal of Object Technology, available online (and for free) at [4]. We organize a summer school, LASER (the acronym for a general description of our work, Laboratory for Applied Software Engineering Research); the first, very successful session took place in September 2004 in Elba (http://se.inf.ethz.ch/laser). The LASER school is the only regular one of its kind, focused on applied software technology. In ETH seminars and courses we have frequent visitors and speakers, many among the best-known names in the field. On October 10-14 of 2005, as part of the ETH 150th-year anniversary celebration, we will hold at ETH a conference on "Verified Software", under the auspices of IFIP, the International Federation for Information Processing; this promises to be a milestone event, showing to the world the steady progress that software verification has made over the past decades.

Software engineering is everywhere. Experts in the field are in constant demand, accompanying the growth of the IT industry and its expansion into ever more areas of human endeavor. The articles in this special issue of Visionen present some of the work done at ETH; we hope that they'll give you some idea of what we are doing to address the challenges of ever bigger and better software. Our Web pages at [5] contain lots of information; if you'd like to know more, we'll be happy to oblige: don't be shy about taking the elevator in IFW, all the way to the top floor.

Links:

[1] http://se.inf.ethz.ch/touch
[2] http://se.inf.ethz.ch/research
[3] http://se.inf.ethz.ch/fats
[4] http://www.jot.fm
[5] http://se.inf.ethz.ch

Software Engineering

Eine kurze Geschichte der Zeit ... nach der ETH

CLAUDE KNAUS - AUF DER WALZ

Sechs Jahre ist es nun her, seit dem ich den etwas kryptischen Titel Dipl.Inf.-Ing. tragen darf. Es ist wohl eher aussergewöhnlich, dass ich nach dieser Zeit bereits meinen 4. Job ausübe. Die unterschiedlichen Arbeitsplätze erlaubten es mir, für High-End Server, Desktop-Maschinen und Embedded-Systemen Software zu entwickeln, Standardisierung von APIs voranzutreiben und Erfahrung mit verschiedenen Firmenstrukturen und Prozessen zu erlangen.

Station 1 - SGI

Bis kurz vor dem Studienende war ich davon überzeugt, dass meine Karriere der Akademie gehörte. Meine Zuneigung zum Visuellem und Ästhetischem führte über die Vertiefungsfächer GDV und Computer Vision zum Praktikum bei SGI. Es kam, dass ich bei derselben Firma meinen ersten Job antrat. Wie viele andere, die damals von SGI begeistert waren, war mir meine genaue Arbeit eigentlich egal. Hauptsache ich hatte einen Job und das erst noch bei der coolsten Computerfirma der Welt!

Damals hiess die Firma noch Silicon Graphics und kein Buchstabe des Logos erinnerte an Homer Simpson. Mein Jobtitel war PSO - Professional Services Engineer, also ein Ingenieur der seine (Softwareetwicklungs-) Dienste für Kunden anbietet. So ein Kunde war bereits definiert, eine norwegische Ölfirma welche mit modernsten Mitteln, d.h., mit Virtual Reality nach Öl in Meersedimenten bohrt. Das Projekt war ziemlich

innovativ - eine Seltenheit für PS Projekte: Es ging darum, in einer VR CAVE Umgebung einen Volume Renderer zu entwickeln, der es erlaubte interessante Teile (= Öl = Geld) in real-time mit einem virtuellen Jedi Lightsaber zu extrahieren. 3 Monate und einige Nächte später war die Tat vollbracht - es konnte dann mit Hilfe einer Drittfirma in die Umgebung des Kunden eingebunden werden.

Später profitierte ich von der weltweiten Verteilung der R&D-Abteilung, und wechselte zu dieser Organisation über, ohne den Arbeitsort wechseln zu müssen. In einer global verteilten Firma wie SGI zu arbeiten, bietet viele Reisegelegenheiten, v.a. in das Silicon Valley. Einerseits war das HQ (heute HQ von Google) dort lokalisiert, andererseits wurde die Produktentwicklung von dort aus geleitet. Als R&D- Mitglied entwickelte ich nun an einem Produkt (OpenGL Multipipe), ein X- und OpenGL-Proxy für die Ansteuerung von Grossbildschirmen wie sie bei der Deutschen Telekom (96 Bildschirme) Anwendung finden.

Station 2 - OTI

Nach 2.5 Jahren entschloss ich mich eine 180 Grad Kehrtwende zu machen. Bei SGI wurde hauptsächlich in C++ programmiert, meist aber eher im Stile von C. Was mich störte war eine Lücke, die ich seit der ETH Zeit nicht hinreichend füllen konnte: OOP. Wie ich mich erinnere, war die erste Java Vorlesung an der ETH hoffnungslos überfüllt, eine andere Vorlesung über Design Patterns kollidierte mit dem Inline-Skating-Dance-Kurs.

Also entschloss ich, mich bei OTI vorzustellen. OTI (Object Technologies International) - heute IBM - war nach Visual Age Micro Edition dabei, Eclipse zu entwickeln. Ich durfte mich auf dem *editor framework* von Eclipse bei der Entwickling neuer Features austoben.

Software zu entwickeln, die man selbst benützt um weitere Software zu entwickeln (man bemerke die Rekursion) hat etwas Befriedigendes. Nebst der Motivation durch den Eigennutzen bietet es den Vorteil, die Software aus der Sicht des Users kennenzulernen. Bekanntlich benötigt gutes *user interface design* viele Feedback-Zyklen.

Mit dem Entscheid von IBM, Eclipse der Open Source Community zu übergeben, änderte sich einiges. Das Durchkämmen der Mailingliste und Beantwortung der Fragen, wie auch die Durchsicht von *feature requests* in Form von *patches* gehörte zu den neuen Pflichten. Auf das Fixen von Bugs oder Implementierung eines Features folgten zahlreiche positive Feedbacks aber auch konstruktive Kritik aus der Community.

Mit Erich Gamma als Manager kam ich nicht um das Schreiben von Unit-Tests herum. Eine gute Disziplin, von der man nicht mehr loskommt. Test-infected eben.

Station 3 - back to the roots

Nach dem zweiten Besuch bei SIGGRAPH in LA holten mich die Gefühle für Computer Grafik wieder ein. Also landete ich wieder bei SGI.

Mittlerweile war die Fabrik mit über 300 Mitarbeitern wegen Restrukturierung geschlossen worden; nur ein kleines Office mit meinen ehemaligen Kollegen blieb bestehen.

Nach einem kurzen Abstecher in den Niederungen der Treiber-Entwicklung hatte ich die Möglichkeit die *conformance tests* für OpenGL ES zu entwickeln. OpenGL ES (ES steht für Embedded Systems) ist ein Subset von OpenGL, das von *The Khronos Group* standardisiert wird. Nur OpenGL ES Implementierungen welche die *conformance tests* passieren, dürfen sich auch mit dem Logo schmücken.

Der Prozess der Standardisierung hat eine eigene Dynamik. Da viele der Firmen direkte

Konkurrenten sind, ist die Projektplanung nicht so einfach wie in einer Firma. Grosse Events/ Kongresse definieren die *dead-line*. Interessant war auch, dass die Zusammenarbeit zwischen den Konkurrenten manchmal besser klappte als mit Mitarbeitern innerhalb einer Firma. Da die Beteiligten über drei Kontiente verteilt waren, fielen Konferenzschaltungen leider oft auf ungewöhnliche Zeiten.

Seit diesem Sommer sind Handys von namhaften Herstellern mit OpenGL ES Implementierungen im Handel erhältlich.

Station 4 - Esmertec

Nach einer weiteren Restrukturierung wurde die entgültige Schliessung unseres übrig gebliebenen SGI Offices beschlossen. Es bot sich glücklicherweise die Möglichkeit, mit der Firma Esmertec das Office wiederzuverwenden. Esmertec ist ein Spin-Off der ETHZ. Einige werden sich sicher an X-Oberon (Oberon mit Real-Time flavor für Robotik) erinnern. Esmertec stellt u.a. Java VMs für zahlreiche Handy Hersteller her.

Hier habe ich die Möglichkeit am user interface kommender Handy Generationen zu entwickeln. Daneben habe ich durch die Erfahrung mit OpenGL ES die Gelegenheit am JCP (Java Community Process) teilzunnehmen um die Standardisierung von JSR 239 (Java Standard Request) und anderen Standardisierungsbemühungen voranzutreiben.

Abschliessendes

Rückblickend gab es eine Vorlesung, die ich als speziell wertvoll erachte: Informatik I von Jürg Gutknecht. Die Vorlesung über Dijkstra's Methodik für die Beweisführung von Programmen sehe ich auch heute noch als Highlight des Studiums. Die Vorlesung hat mir gezeigt, dass Programmieren methodisch angegangen werden kann. Ich kam zur Einsicht, allerdings erst nach einigen Jahren praktischer Erfahrung, dass dies auch für Softwareentwicklung und -architektur im Allgemeinen gilt.

Neben dem technischen Aspekt ist Wissen über Projektmanagement sehr nützlich, wenn nicht unabdingbar, sowohl für das Management der eigenen Arbeit, wie auch für die Zusammenarbeit und Kommunikation mit Produktmanagement und Marketing. Ein guter Manager (zum Glück war ich mit solchen gesegnet) kann letzteres enorm vereinfachen. In einem zukünftigem Vorstellungsgespräch würde ich den Manager genau so interviewen wie er mich interviewen würde.

Software Engineering

Die letzte Woche im Leben eines SE Studenten

BENNO BAUMGARTNER – STERNZEICHEN GEEK

Da ich, so behauptet zumindest Till, der einzige Geek unter den SE Studenten bin obliegt mir die Ehre, das SE Departement aus der Sicht eines Studenten vorzustellen.

Am besten ich beschreibe euch, was ich während einer Woche treibe, und da eine solche Woche äussert Ereignislos ist, reichen dafür zwei Seiten locker. Diese Woche ist eine ganz besondere für mich: Es ist meine letzte.

Am Montag Abend halten Reto und ich einen Vortrag über Grid-Computing an der Technischen Berufsschule in Zürich. Jeder, der Informatik Didaktik 2 besucht, muss einmal unterrichten.

Am Dienstag Morgen überlegen wir Info4 Assis wie wir die Studenten diese Woche am besten Quälen können.

Ich mache jetzt schon das 3. Semester Hilfsassi und kann das eigentlich nur jedem empfehlen: Man lernt eine Menge Leute kennen und es gibt auch noch Geld dafür.

Am Nachmittag präsentieren Thomas und ich unsere Lösung für eine verteilte Matrixmultiplikation, geschrieben mit SCOOP [1].

Danach ist Arnaud Bailly, a.k.a. die Pi-Calculus Maschine, wieder an der Reihe. Pi-Calculus ist eine Erweiterung von Lambda-Calculus mit Input- und Outputkanälen. Man kann damit Concurrency mathematisch beschreiben. Wer auf Hardcoremathe steht sollte sich die COOP Vorlesung unbedingt antun, wer es eher gemütlich mag, sollte die Finger davon lassen.

Am Mittwoch Morgen ist dann Informatik Didaktik bei Prof. Werner Hartmann. Die Vorlesung ist sicherlich ein Highlight der Woche. Ich kann das Nebenfach Didaktik nur jedem Empfehlen: Man sieht mal ein bisschen über den eigenen Bildschirmrand.

Ein anderer Höhepunkt (nein wirklich) ist Prof. Müllers Vorlesung: Semantik von Programmiersprachen. Es geht darum die Semantik einer Programmiersprache mathematisch zu

beschreiben, um dann verschieden Eigenschaften, die die Sprache haben sollte, beweisen zu können. Die Thematik ist nicht einfach, aber Peter Müller schafft es, die Dinge zu erklären, ohne sich in mathematischen Details zu verrennen, auch wenn der Bogen bei der Fixpunktiteration hörbar zu knacken begann.

Am Nachmittag habe ich dann 4 Stunden Professional English. Es ist eine Tortur. Da ich am 1. August für mein Praktikum zu Eiffel Software [2] nach Santa Barbara gehe, sollte ich ein paar Worte Englisch lernen.

Nein, nein, ich beschäftige mich nicht nur mit Fixpunktiterationen, manchmal trinke ich auch (natürlich nur mit äusserstem Wiederwillen) ein, zwei Bierchen, z.B. wenn EM ist.

Am Freitag Nachmittag dann was ganz besonders: Die Info4 Studenten stellen ihre Spiele [3], die sie die letzten vier Wochen mit ESDL [4] geschrieben haben, vor.
Wir sind alle beeindruckt von dem Gezeigten. Teile von ESDL habe ich als Semesterarbeit geschrieben (es gibt noch viel zu tun, Bewerbungen an Till G. Bay).

SEler beschäftigen sich also nicht nur mit Patterns, vieles ist tatsächlich reine Mathematik. Warum also einen SE Master machen? Ganz einfach, SEler haben den besten Überblick.
Machts gut liebe Leut und vergesst das wesentliche niemals: DIE VIEW ENTHÄLT NICHT DIE DATEN!

Links

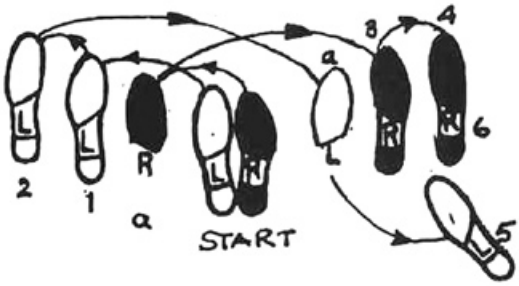
[1] se.inf.ethz.ch/research/scoop.html
[2] www.eiffel.com
[3] se.inf.ethz.ch/people/bay/esdlgames/
[4] eiffelsdl.sourceforge.net

Software Engineering

Yes Sir, I can Boogie (and Jive)

ÁDÁM DARVAS AND PETER MÜLLER - INFORMATIKER MIT HÜFTSCHWUNG

Software is correct, if it is free from coding errors (e.g., null-pointer dereferencing) and satisfies its specification. In industrial software development, testing is the prevalent means to come at least close to correctness. However, testing can reveal bugs in a program, but can never assure that the program is free of bugs. Every software user knows the limits of testing and how buggy even commercial software often is.



Research has been advocating formal program verification as an alternative and complement to testing. Relevant program properties can be specified and implementations can be proved to be correct with respect to their specification. In the past, verification had very limited practical applications, usually in the development of safety- and security-critical systems where malfunction of the system can have serious consequences, for instance, loss of human life.

Over the past few years, tremendous progress has been made in the development of specification and verification techniques, and many experts think that formal methods are finally becoming widely applicable. Key to such a breakthrough are powerful tools that relieve software developers (who are usually not experts is formal verification) of carrying out proofs themselves by automating significant parts of the verification effort.

The Software Component Technology Group at the Chair of Software Engineering works on two such tools, called Jive and Boogie, and on the underlying specification and verification techniques for object-oriented programs.

Jive

In cooperation with the University of Kaiserslautern, our group works on the Java Interactive Verification Environment, Jive. Jive allows one to interactively prove that Java programs fulfill their specifications.

An important property of Jive is user interaction. Since program verification is usually con-

cerned with undecidable properties, verification tools cannot prove correctness automatically. Jive uses strategies to automate parts of proofs. Whenever the tool does not know how to continue, it prompts the user for help. Thereby, programs can be verified in a semi-automated way: The user can focus on the interesting and difficult parts whereas the tool takes care of boring, hence error-prone, proof steps.

Currently the tool is being re-implemented in order to change the specification language from first-order logic to JML (Java Modelling Language), a specification language that is specifically tailored to Java. This change will ease the task of writing specifications for programmers who do not have a solid background in mathematics or logic.

Jive is our platform for experimenting with new techniques and for evaluating our work in case studies. In the near future we also plan to use Jive to teach semantics of programming languages and formal methods.

Boogie

Modularity is a key concept in software engineering to make the development of large applications feasible. Modules can be written, compiled, end tested independently of each other. One of the most important benefits of modularity is reuse. Modern languages come with huge class libraries that simplify software development drastically. However, whereas modularity is nowadays a matter of course for syntactic properties such as typing, modular verification still poses difficult research problems. Especially object-oriented language features such as inheritance and dynamic method binding are difficult to handle and often lead to surprising behavior.

In cooperation with Microsoft Research, we develop modular verification techniques that can be applied to Java and C#. These techniques are implemented in the verification tool Boogie, which is currently being implemented at Microsoft Research. Boogie is fully automated. The tool tries to prove program properties automatically. For all properties that cannot be proved, runtime checks are added to the code to support systematic testing. This combination of verification and testing will enable a smooth integration of Boogie in the industrial software development process.

Student Projects

The development of Jive constantly offers a number of possible Semester-, Master-, and Diploma-projects. These can range from theoretical issues to more implementation based projects. Here we give an example to both kinds of projects.

- JML enables one to write precise and abstract specifications by the use of so-called specification-only model classes that provide a mathematical model for data types such as sequences, sets, etc. In order to faithfully support these model classes in Jive, they have to be mapped to Jive's internal logic, which is based on the Isabelle theorem prover. This mapping has to be precisely defined and implemented in Jive.
- The usability and practicality of a program prover highly depends on its user interface since users often have to interact with the tool. Currently Jive's GUI displays proof states in a way which is only comprehensible to someone who is familiar with the internal data structures of the tool. In the future we would like to change this so that proof states are represented in a way which is easily comprehensible to Java programmers.

Link

http://www.sct.inf.ethz.ch

Software Engineering

This object is mine, Mine, MINE!!

WERNER DIETL AND PETER MÜLLER - GIERIGE INFORMATIKER

Large object-oriented programs can become quite complicated, which makes them difficult to write, understand, and maintain. One of the complications is that all objects of a program execution potentially reference and modify each other. The Software Component Technology Group at the Chair of Software Engineering investigates ways to reduce the complexity of object-oriented programs. We are especially interested in new language features that allow programmers to structure the heap memory and to use this structure to reason about programs. In this article we describe our work in this area.

Aliasing occurs in object-oriented programming whenever an object is referred to by more than one object or variable. For example, the cells in a doubly-linked list are, in general, referenced by the predecessor, the successor, and possibly by the list head and iterators.

Aliasing is a fundamental principle for the flexibility and efficiency of object-oriented programs, but in some cases it is undesirable and a source of errors: the existence of aliases can lead to unforeseen manipulations of object structures, leading to inconsistencies and violations of invariants.

Werner Dietl **Peter Müller**

Unintentional aliasing can also be a security problem. For example, in the Java SDK 1.1 the concept of class signing was introduced. An implementation similar to the following allowed applets to get information on all the signers of a class.

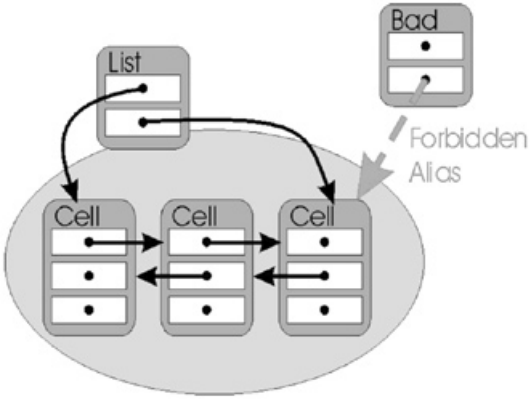
```
class System {
  private ClassSigner[] signers;

    public static ClassSigner[] get-
ClassSigners() {
     return signers;
   }
}
```

It was also possible to get information on which signers are trusted. Classes with trusted signers are allowed to perform privileged actions such as accessing the file system. But now it was easily possible to circumvent the security mechanism: The following code, which could be part of a malicious applet, manipulates the array of class signers to add a trusted signer. Thereby, it gets the permission to do privileged actions.

```
class Violator {
  void m() {
    ClassSigner[] mysigners = System.g
etClassSigners();
     mysigners[0] = System.getValidSig
ners()[0];
    doPrivileged();
  }
}
```

The problem with this code is that getClassSigners returns a reference to an internal data structure. Although the field signers is private, and thereby protected from direct access, a reference to the internal array can be returned by the function and then used externally to modify the representation of the object.



Readonly References

What the Java developers did to fix the problem is to return a copy of the internal array instead of a direct reference. Now classes can read the copy of the data, but their changes are not reflected in the system. However, copying data structures causes performance overhead. Moreover, it is difficult to keep the copies up-to-date. What would be nice to have is the possibility to allow other objects to read data, but prevent them from modifying it.

In current mainstream programming languages references give full access to an object and there is only limited support for the programmer to express that some references should only be used to read data. Many systems therefore rely only on informal documentation to express which objects are allowed to be modified by a method. Being able to check these properties by a compiler would ease development and increase confidence in the system behavior. Problems like the security bug in Java could easily be avoided.

Our Work

The Software Component Technology group works on language features that give programmers fine-grained control over references and their usage. To allow compilers to check statically how references are used, we express aliasing properties as type information. With the Universe type system we developed a flexible mechanism for alias control and readonly references to objects.

The Universe type system uses the concept of object ownership to express that one object is owned by another object and therefore part of its internal representation. For instance, the Universe type system can express that the array of signers in the above example is owned by a system object. Representation objects are encapsulated within the owning object and can only be modified through the owner. In particular, our type system does not

allow a method to return a non-readonly reference to a representation object. That is, the compiler would report an error for method getClassSigners because it exposes the internal representation to other objects.

To allow the safe sharing of information, our language supports readonly references, which can only be used to read information from objects, but all modifications are prohibited. Readonly references allow the efficient sharing of information without worrying about modifications.

We implemented these concepts in a research compiler for Java and also integrated it into the Java Modeling Language JML, a specification language that allows one to describe program behavior formally (similar to Eiffel contracts). This allows the large research community that uses JML to also apply the Universe type system in their developments.

Another application area we are interested in is Java-enabled smart cards. Using the Universe type system it is possible to find certain safety problems statically that, if undetected, would result in runtime errors. Runtime errors can be extremely costly in this case, because it means that the smart card has to be replaced to fix the error.

Student Projects

Advanced type systems like the Universe type system are a very active research area. We try to improve their expressiveness, make their application easier, find new application areas, and evaluate their practicality in case studies. Most of these activities entail interesting topics for semester, master, and diploma projects including:

- Fundamentals of the Universe type system, e.g. extensions to improve the expressive power, work on the formalization of the type system, type inference
- Implementation work, e.g. extensions of the Universe compiler, additional tools, IDE integration, visualization
- Applications of the type system, e.g. case studies, tailoring the type system for a specific problem

The ongoing development of the Universe type system always raises new project topics. Contact us if you are interested in working with us! Who knows, maybe your work will influence the next generation of object-oriented languages.

Link

http://www.sct.inf.ethz.ch/

werbung von Binkert

Videosessions

TILL - WÄRE GERN BRAD PITT

Mit Freude und Stolz kann ich euch hier das Programm der Videosessions im WS 04/05 präsentieren. Ungefähr alle zwei Wochen wird im IFW A36 ein Film gezeigt. Im Rahmen der Weihnachts- Wunschvideosession habt ihr auch wieder die Möglichkeit, mitzubestimmen,was gezeigt wird.

24. November, *Fight Club*

Man nehme einen durchschnittlichen Mann mit einem durchschnittlichen Job, der ein durchschnittliches Leben in einer durchschnittlichen US-Grossstadt führt. Diesen setze man im Flugzeug neben einen kuriosen Seifenverkäufer, und schon hat man eine Geschichte, die sich im wahrsten Sinne des Wortes gewaschen hat. Um Seife geht es nämlich, um die Erika Pekkari Tagesdecke von IKEA, um eine ketterauchende Frau, um harte und weniger harte Kerle und um die ganz grosse Verschwörung. Die namenlose Hauptfigur des Films (Edward Norton) trifft auf den charismatischen Seifenverkäufer Tyler Durden (Brad Pitt) und gründet mit ihm zusammen den Fight Club, einen Kämpferzirkel, in dem Männer ihre angestauten Aggressionen ablassen können, um ihren Seelenfrieden zu finden. Doch je mehr er sich dem Abgrund zur Apokalypse nähert, desto mehr entgleitet ihm die Kontrolle. Und Tyler Durden scheint sein eigenes Spiel zu spielen...

Der Regisseur David Fincher ist bekannt für seine knallharten und spannenden Filme (“Panic Room”, “Se7en”, “The Game”) und führt die Zuschauer auch in Fight Club des öfteren auf falsche Fährten. Die illustre Schauspielercrew (u.a. auch Meat Loaf, unbedingt sehenswert!) und das unerwartete Ende lassen Thriller-Fans auf ihre Kosten kommen.

8. Dezember, *Elling*

Im Gegensatz zu vielen (Hollywood-)Produktionen kam im Jahr 2001 wieder mal eine gut erzählte Komödie aus Norwegen in die Kinos. Elling und Kjell Bjarne sind ein klassisches Filmpaar. Der Grobschlächtige und der Feine, der Grosse und der Kleine, der Sorglose und der Gedankenversunkene. Elling ist vierzig Jahre alt, etwas pedantisch und ein Muttersöhnchen. Nach Jahren in einer psychiatrischen Anstalt mit seinem Zimmergenossen Kjell Bjarne muss er sich nun zum ersten Mal im richtigen Leben zurechtfinden, was nicht immer einfach ist. Ist es nicht unnatürlich mit einem Plastikdings in der Hand dazustehen und mit einem Menschen zu reden, den man nicht einmal sehen kann? Und Kjell

Bjarnes grösste Sorge ist, dass er das halbe Leben schon hinter sich hat und noch nie gebumst hat.

Zusammenfassend, ein schöner, herzlicher Film.

22.Dezember,
Weihnachts-Wunschvideosession

Was gibt es schöneres als eine Videosession auf Weihnachten zu bekommen? (Das war eine rhetorische Frage.) Ihr werdet dieses Jahr auf jeden Fall eine bekommen. Die Filme, aus denen ihr dann euren Lieblingsfilm auswählen könnt, stehen noch nicht fest, Vorschläge nimmt videosessions@ vis.ethz.ch entgegen. Die Entscheidung wird, wie immer, in einer Umfrage im Forum fallen, welche euch rechtzeitig per Email und über die VIS-Homepage angekündigt wird.

12. Januar, *Amores Perros*

Ein Film mit drei Geschichten, die immer mal wieder aufeinander treffen, wie zum Beispiel bei einem Autounfall, den man immer wieder aus einer anderen Perspektive sieht. Octavio (Gael Garcia Bernal), will mit illegalen Hundekämpfen und dem Hund seines Bruders an das schnelle Geld kommen. El Chivo (Emilio Echevarria), ist ein ehemalligen Guerilla-Kämpfer, lebt nun mit seinen Hunden auf der Strasse und verdient sein Geld als Auftragskiller. Und schliesslich das berühmte Model Valeria, die sich nach einem Autounfall mit dem Rollstuhl und damit, dass Fifi unter dem Parkett verschwindet, abfinden muss.

Dem Regisseur Alejandro Gonzàlez Inarritu brachte sein letzter Film «21 Grams» etwas mehr Publicity ein, aber schon Amores Perros wurde zurecht für einen Oscar nominiert.

2. Februar,
Monty Python and the Holy Grail

Ich wage mal zu sagen, das ist ein Klassiker der Filmgeschichte, zumindest wenn man schwarzen Humor mag. Die Handlung des Films ist eigentlich Nebensache, und nur ein Gefäss für die unzähligen Monty-Python-typischen Gags. Trotzdem kurz zur Handlung: Es geht um König Arthur, der sich seine Ritter der Tafelrunde zusammensucht und zum Schloss Camelot pilgert, wo sie dann von Gott aufgefordert werden, sich auf die Suche nach dem heiligen Gral zu begeben. Auf dem Weg machen sie dann Bekanntschaft mit dem schwarzen Ritter, dem Franzosen, dem Hüter der Brücke, dem Killerkaninchen, den “Rittern, die immer Ni sagen”, der heiligen Handgranate, und so weiter. Wer den Film kennt, weiss von was ich rede, und die anderen sollten einfach an die Videosession kommen. :)

Gewinnspiel --> Seite 54


foo.bar
struct Non_Alcoholic {
class LONGDRINKS inherits ALCOHOL {
End

[i][A][E][T][H]

Informatik-Alumni ETH Zürich

An alle Informatikstudierenden

Dies ist der zweite Artikel einer Serie, mit welcher wir euch die Informatik Alumni ETH (IAETH) näher verstellen. Als «VIS der Ehemaligen» fördern wir den Kontakt unter unseren Mitgliedern und euch. In Zusammenarbeit mit dem VIS bieten wir über unser Mitgliederportal IAETHOnline (ww.iaeth.ch) eine Suchfunktion, anhand derer Ihr die freigeschalteten Mitglieder-CVs durchsuchen könnt. Interessante Mitglieder können dann direkt kontaktiert werden. So erhältst du zuverlässige Information über verschiedene Firmen und Arbeitgeber deiner Wahl, oder Tipps und Tricks zur Karrieregestaltung. Im geschützten Mitgliederbereich unserer Website ww.iaeth.ch publizieren wir beispielsweise Portraits von ausgewählten Mitgliedern. Als exklusive Dienstleistung für die Informatikstudierenden der ETH Zürich geben wir in loser Folge einige dieser Portrait Letters wieder. In diesem Zusammenhang stellen wir euch nach Thomas Kistler heute Dominik Gruntz vor, der Professor an der Fachhochschule Aargau in Brugg-Windisch wurde:

BEGIN PORTRAIT DOMINIK GRUNTZ

Seine ersten Erfahrungen waren eher ernüchternd. Als Dominik Gruntz Ende der siebziger Jahre im Gymnasium den Computerkurs absolvierte, konnte er dem Fach nur wenig abgewinnen. Das Programmieren mit Basic begeistere den Schüler überhaupt nicht. Seine Leidenschaft für Software entdeckte Gruntz erst später an der ETH Zürich, als ihm im ersten Semester Elektrotechnik das Fach Informatik am besten gefiel. Die Studienrichtung zu wechseln, war offenbar ein richtiger Entschluss - immerhin wurde er von der ETH für seine Diplomarbeit mit der Silbermedaille ausgezeichnet. Allzu viel Bedeutung misst er dieser Ehrung heute allerdings nicht bei. „Ich weiss, wie solche Sachen zustande kommen“, meint er lakonisch.

Der Entscheid, eine Dissertation zu machen, erfolgte dann ein Stück weit auch aus Bequemlichkeit, erklärt Gruntz. „Ich spielte damals zwar noch mit dem Gedanken, ein Zweitstudium in Mathematik zu absolvieren. Doch ich hätte da von ganz vorne anfangen müssen, und das wollte

ich nicht.“ Seine Affinität zur Mathematik konnte er auch so ausleben, arbeitete er doch bei der Entwicklung des Mathematik-Programms „Maple“ mit. Die Arbeit im Grenzgebiet zwischen Informatik und Mathematik war zwar anregend, aber auch schwierig. „Irgendwann merkte ich, dass mir das mathematische Grundwissen fehlt. Zudem fühlte ich mich als Grenzgänger von den beiden Communities nicht recht ernst genommen.“

Nach dem Abschluss der Dissertation arbeitete Gruntz bei der Firma Oberon microsystems, die Studienkollegen von ihm gegründet hatten. Ziel der Spin-off-Firma ist es, Kunden beim Entwickeln von Komponenten-basierter Software zu unterstützen. Dazu setzt die Firma ihre eigene Programmierumgebung ein, die sich am Oberon-System von Niklaus Wirth orientiert. In dieser Zeit entdeckte Gruntz sein Talent als Lehrer. „Ich gab bei unseren Kunden Schulungen, und dabei merkte ist, dass ich es verstehe, komplizierte Sachverhalte einfach darzustellen.“

Das Vermitteln von Stoff fasziniert ihn noch heute. „Ich bin gerne Lehrer“, meint Gruntz. An einer Mittelschule zu unterrichten, könnte er sich aber nicht vorstellen. „Mein Ziel war es stets, an einer Fachhochschule (FH) zu arbeiten.“ So war es ihm nur recht, dass er noch während seiner Zeit bei Oberon Lehraufträge bei verschiedenen Schulen übernehmen konnte.

Den Sprung zum fest angestellten Lehrer schaffte er schliesslich 1999. Heute ist Gruntz Professor für Informatik an der FH Aargau in Brugg-Windisch. Seine Wahl fiel in eine turbulente Zeit, als aus der HTL Brugg-Windisch die Fachhochschule Aargau wurde. Immerhin bot das wechselhafte Umfeld auch Chancen und ermöglichte ihm, eigene Ideen zu verwirklichen. „Der Studiengang Informatik - übrigens der älteste in der Schweiz - war in Brugg damals noch stark Hardware-orientiert. Dies schien mir nicht mehr zeitgemäss, und ich regte daher eine Neukonzipierung der Ausbildung an.“

Heute arbeiten die Schulen der Fachhochschule Nordwestschweiz in der Informatikausbildung eng zusammen. Das zweijährige Grundstudium ist koordiniert, und darauf aufbauend bieten die Teilschulen unterschiedliche Vertiefungen an. „Diese Zweiteilung ist sinnvoll, weil die Studenten am Anfang ja nicht wissen, in welche Richtung sie sich spezialisieren sollen“, erklärt Gruntz. Das Konzept scheint anzukommen; jedenfalls entwickeln sich die Studentenzahlen erfreulich. Und das ist nicht selbstverständlich: „Im Gegensatz zu anderen Fächern haben unsere Studenten häufig einen anderen Beruf gelernt, und ein Elektrozeichner beispielsweise überlegt es sich heute zweimal, ob er wirklich in die Informatikbranche wechseln will.“

Nebenbei hält Gruntz auch noch an Universitäten Vorlesungen, zur Zeit auch an der ETH Zürich. Das ermöglicht ihm, Fachhochschule und Universität direkt zu vergleichen. „Es gibt schon deutliche Unterschiede“, stellt er fest. „So zeigen sich die Studenten an der ETH anfänglich immer irritiert, wenn ich sie zu sehr mit Fragen in den Unterricht einbeziehen will.“ Und wie steht es mit den Berufsaussichten? „Die FH-Abgänger sind sicher praxistauglich. Sie haben mit Werkzeugen gearbeitet, die sie in der Industrie brauchen können. Das ist an der ETH etwas anders. Dort müssen sich die Professoren in der Forschung profilieren und können nicht einfach ein Projekt auf einem Industriestandard aufbauen.“ Doch das muss für die Studenten kein Nachteil sein. „Die ETH-Absolventen eignen sich das fehlende Wissen in der Regel rasch an. Und wenn es darum geht, Konzepte zu entwickeln, kommt ihnen die abstrakte Denkweise zugute.“

Das Arbeiten an einer Fachhochschule gefällt Gruntz. „Ich habe Freude am direkten Kontakt zu den Studierenden. Und wenn ich merke, dass sie plötzlich etwas begreifen, dann gibt mir das Befriedigung." Geradezu paradiesisch findet Gruntz, dass er an der FH das Verhältnis Forschung zu Lehre flexibel gestalten kann. „Es gibt Kollegen, die sind zu hundert Prozent Lehrer, und es gibt solche, die machen zu achtzig Prozent Forschung." Im Gegensatz zur Lehre muss die Forschungstätigkeit allerdings über Projekte finanziert werden. „Wir können hier sicher keine Grundlagenforschung betreiben, sondern beschäftigen uns mit praxisorientierten Problemen."
Gruntz selber arbeitete zusammen mit einem Assistenten an einem Projekt im Auftrag der Firma Hectronic. Diese will ein System entwickeln, mit dem Parkgebühren via Handy bezahlt werden können. „Wir haben zuerst in einer Semesterarbeit eine WAP-Applikation entwickelt. Da sich diese als zu umständlich erwies, haben wir nun eine Java-Lösung erarbeitet." Das Projekt ist im Moment allerdings blockiert. „Ein Pilotversuch in Österreich ist aus politischen Gründen gescheitert. Und in den anderen Ländern kämpfen wir mit Patentproblemen. Daher macht es im Moment keinen Sinn, die Applikation weiter zu entwickeln."
In Zukunft wird Gruntz jedoch weniger Zeit zum Forschen haben, wird er doch ab August 2003 die Leitung des Studienganges Informatik an der FH Aargau übernehmen. „Im Hinblick auf die Umsetzung der Deklaration von Bolognia mit den Bachelor- / Master-Studiengängen wird sich die Ausbildung an unserer Schule grundsätzlich ändern", erklärt er. „Allerdings möchte ich weiterhin an einer soliden fachlichen Grundausbildung festhalten."
In Zukunft möchte Gruntz aber auch Kurse für die Praxis anbieten. „Die Weiterbildung wird für ausgebildete Informatiker zunehmend wichtig. Die Entwicklung geht ja weiter, trotz der gegenwärtigen Krise, und als Fachmann wird man immer wieder mit neuen Begriffen konfrontiert, die in das Weltbild eingeordnet werden müssen." Ein Kurs kann da sehr hilfreich sein, um sich auf den neusten Stand zu bringen, erklärt Gruntz. „Vielleicht auch einfach nur um zu merken, dass sich hinter den neuen Schlagwörtern nicht unbedingt auch neue Ideen verbergen."
Zur Person
Dominik Gruntz, Jahrgang 1964, begann 1983 an der ETH Zürich Informatik zu studieren. Nach seiner Promotion im Jahre 1994 arbeitete er während mehr als vier Jahren bei der Firma Oberon microsystems AG in Zürich. Seit 1999 ist er Professor für Informatik an der Fachhochschule Aargau / Nordwestschweiz in Brugg-Windisch. Zu seinen Hauptinteressen gehört die Entwicklung von Komponenten-Software. Er lebt zusammen mit seiner Frau und seinen drei Kindern in Wettingen. Kontakt: d.gruntz@fh-aargau.ch

Wir hoffen, euch damit einen kleinen Einblick in die faszinierenden beruflichen Möglichkeiten für Informatik-Absolventen gegeben zu haben, und stehen euch jederzeit für weitere Fragen zur Verfügung.

Für den IAETH-Vorstand: Markus.Grob@iaeth.ch

SA/DA Shortcuts

SA/DA/Master

Projects available in Laboratory for Software Technology (LST)

LABORATORY FOR SOFTWARE TECHNOLOGY

The Laboratory for Software Technology is a research group in the Computer Science Department of ETH Zürich dedicated to research in all aspects of software construction: software design and implementation, programming methodology, and performance evaluation. Here you find a summary of student projects (for the complete list visit: http://www.lst.inf.ethz.ch/teaching/sada/index.html.

Stereoscopic Imaging Projects

- These projects will be of interest to students who intrigued by stereoscopic imaging. If you were not able to take CS-251-230 (Stereoscopic Imaging) during the Summer 2004 semester, this is an ideal opportunity to get involved. For those students who took the course, these projects will provide opportunities to enrich their experience and to become more deeply involved in a variety of projects that fall into three basic categories: stereoscopic video capture; presentation of stereoscopic images and films; and stereoscopic video post-production. Students in all academic disciplines are encouraged to participate. Students will work individually on projects and may also work together in a larger group, typically when we create a new capture or display system. At such milestones we often create a stereoscopic film. These projects provide opportunities for Semester, Diploma and Masters level research.
- **Areas of Interest:**
  Computer human interaction and instruction
  Stereoscopic imaging and human-visual perception
  Software development and image processing
  Performance evaluation and storage management
  Electro-mechanical and optical design and construction
  Real time computer control and data capture
- **Contact:**
  Dr. Cary D Kornfeld, 63 27923, RZ H6,
  cary.kornfeld@inf.ethz.ch

Routing in (Large) Ad Hoc Networks (with cars)

- An ad hoc network is formed by a number of wireless mobile nodes (hosts) without any centralized administration. Each node participates in an ad hoc routing protocol that allows it to discover different paths through the network to any other node. You will have an opportunity to improve existing routing protocols or design a new one and then test the results of your work on the real road maps of Switzerland. These scenarios-maps are obtained from the

car traffic simulator developed at ETH Zurich. The following projects are available in this research area (contact Valeri Naoumov, 63 20672, RZ H3, naoumov@inf.ethz.ch):

1. Effective warnings propagation among cars

Using the potential of ad hoc networks to exchange information between cars on a highway can be very attractive: e.g., an ad hoc network may be used to transmit warnings about traffic jams, accidents, construction sites, etc. This message could force a GPS system to recalculate the route to avoid the problem or to warn the driver about the danger. The goal of this work is to implement a reliable flooding protocol for effective warnings propagation in highway-like scenarios.

2. Internet access from cars

Long paths in a dynamic ad hoc network have short life-times and quickly become invalid, especially when cars are used as wireless hosts. Planting base stations (BS) along streets in a city will allow Internet access in cars and will decrease the average path length — data sent from a car need to travel only to the nearest BS, which has "unlimited" wired bandwidth. The goal of this project is to extend the AODV protocol for effective car-BS-car and car-BS-Internet communication.

Resource Management and Wireless Networks

1. Effective TCP Congestion Recovery

TCP can experience dramatic performance degradation when the congestion control window is small. Scenarios that suffer include fetching web pages from a busy web server and transferring files over multi-hop wireless networks. The goal of this project is to design and evaluate new mechanisms to improve the efficiency of TCP congestion recovery under a tiny window condition, by reducing coarse timeout and improving the effectiveness of fast retransmit/fast recovery.

2. Channel-Aware Traffic Analysis on Wireless LAN

Current network packet analyzers (such as tcpdump, ethereal) have limited support for IEEE 802.11 wireless networks (WLAN). They only provide access and analysis of data packets and some of non-data (management, beacon) packets. They do not provide information about channel conditions which is critical to understand the behavior of wireless networks. The tasks of this project include: 1. Programming of the proposed wireless card to get desired channel condition information. 2. Integration of the information about channel conditions into packet analysis tools.

Contact Yang Su, 63-27320, RZ H12, ysu@inf.ethz.ch

Adaptive Main Memory Compression

- The basic idea of a compressed-memory system is to set aside some memory and use it to hold pages in compressed format. By compressing some of the data space, the effective memory size available to the applications is made larger and disk accesses are avoided.
- Because the applications that are good candidates to benefit from main memory compression are those with large data sets, we enable compression only for applications with working sets that exceed the physical memory size. When a large application is executed, the adaptive system strives to find the optimal size of the compressed area by shrinking or growing the compressed area size while the application executes. For applications that execute multiple times, we can lower the adaptivity overhead by recording the optimal size of the compressed area when the applications

execute for the first time. Then, for subsequent executions, the system can set the compressed area size to be equal to the recorded optimal size.

- The goal of our projects is to implement a tool to help the system decide which applications to compress and which not. The tool will monitor the system usage in terms of memory usage, and will record the applications with memory requirements above a certain threshold. When such an application executes, the system will be informed that this application is a candidate for compression and its data will be compressed. The tool will also inform the system whether the application has been executed before and if so, how large was its compressed area size (contact Irina Chihaia, 63-27354, RZ H12, chihaia@inf.ethz.ch

StudentAktiv

Praktikum bei der AdNovum Informatik AG

MANUEL GRABER - ZURÜCK AUS DEM PRAKTIKUM

Auch ich möchte euch die Erfahrungen, die ich im Praktikum bei der AdNovum gesammelt habe, nicht vorenthalten. Hier die Kurzversion: Es hat sehr viel Spass gemacht, ich habe eine Menge gelernt, ich würde es jederzeit wieder tun.

Die AdNovum ist spezialisiert auf die Entwicklung von E-Banking- und Sicherheitslösungen und Applikationssystemen im Finanzdienstleistungs- und Bundesbereich. Zu ihren Kunden gehören bekannte Firmen wie UBS, PostFinance, Swisscom, Telekurs und das EJPD. Gegründet wurde die AdNovum vor 16 Jahren. Mittlerweile hat das Unternehmen über 100 Mitarbeiter und unterhält neben dem Hauptsitz in Zürich Büros in Bern, den USA und Ungarn.

Die AdNovum arbeitet projektorientiert, d.h. es werden keine fertigen Produkte verkauft, sondern kundenspezifische Lösungen entwickelt. Diese Lösungen basieren hauptsächlich auf den von der AdNovum entwickelten Middleware-Komponenten oder Applikationsframeworks wie zum Beispiel Nevis Web (eine Lösung zur Web-Integration), ISI (eine Corba-/J2EE-Implementierung) oder SecStack (eine Security Middleware).

Am Anfang des Praktikums fand die so genannte Safari statt, bei der man alle Mitarbeiter, die in einer Schlüsselposition tätig sind, kennen lernt. So erfährt man, wer für was zuständig ist und an wen man sich später mit Fragen wenden kann. Sitzungen finden in der Regel an der Kaffeebar statt, wo man sich auch kostenlos mit Getränken eindecken kann. Zu Mittag essen kann man zu einem fairen Preis in der hauseigenen „Guccinetta“, die eher ein Restaurant als eine Kantine ist.

Der Security Stack (SecStack)

Meine Praktikumsaufgabe lag im Bereich des SecStack. Dieser bietet ein API, das es erlaubt, einen sicheren Kontext mit einem Kommunikationspartner aufzubauen und innerhalb dieses Kontexts verschlüsselte und signierte Daten zu übertragen. Dieses API ist als GSSv2 (Generic Security Service, RFC2743) standardisiert. GSS funktioniert unabhängig von der eigentlichen Übertragung, indem es nach jedem Aufruf einer Funktion ein «Token» zurückgibt, das man dann dem Kommunikationspartner schicken muss. Da der SecStack auf die verschiedensten Plattformen portiert werden sollte, erfolgte die Implementierung in C (ohne ++). Dabei wird ein Open-Source Abstraktionslayer eingesetzt, der die Eigenheiten der verschiedenen Plattformen kapselt.

Die Praktikumsaufgabe

Zur Qualitätssicherung wurde eine umfangreiche Testinfrastruktur für den SecStack aufgebaut. So gibt es Nightly Builds, mit denen jede Nacht das ganze Projekt gebaut und mit Hilfe von Unit Tests getestet wird. Die Entwickler erhalten dann jeweils am Morgen eine E-Mail mit dem Resultat dieser Nightly Builds. Falls also eine Änderung einen negativen Effekt gehabt hat, sieht man das spätestens am nächsten Morgen. Daneben gibt es einen Weekly Build, der nach dem gleichen Schema abläuft, aber eben nur einmal pro Woche ausgeführt wird. Die Idee dahinter ist, dass man im Weekly Build Tests laufen lassen kann, die länger dauern (z.B. umfangreiche Integrations- oder Lasttests).

Nun sollte ein Tool geschrieben werden, mit dem man Client-Server Tests ohne grossen Konfigurationsaufwand auf mehrere Maschinen verteilen und automatisiert durchführen kann. Meine Aufgabe war die Entwicklung dieses Tools, das sich nun Distributed Test Framework (DTF) nennt. Nachdem die Grundfunktionalität, die das DTF haben sollte, festgelegt war, war ich sehr frei in der Entwicklung und konnte es nach eigenen Ideen implementieren. Meine Betreuer und die übrigen Software-Entwickler standen mir dabei immer hilfreich zur Seite.

Das Resultat

Das DTF besteht aus drei Teilen: einem Runner, einem Agenten und einer Bibliothek, die den auszuführenden Testcode enthält.

Der typische Ablauf eines Tests sieht so aus, dass der Runner ein Konfigurationsfile einliest und gemäss den Informationen in diesem File einen oder mehrere Agenten startet, die ihrerseits die Bibliothek laden. Der Testcode in der Bibliothek wird ausgeführt und am Schluss wird das Ergebnis dem Runner mitgeteilt. Um den Output während eines Tests aufzuzeichnen, hat jeder Agent ein Logfile, das er nach Abschluss der Tests dem Runner übermittelt. Dabei kann im Konfigurationsfile angegeben werden, wie viele Agenten auf welchen Rechnern gestartet werden sollen, und welche Tests diese ausführen sollen.

Ich programmierte also in C und damit konnte ich mich mit (in Zeiten von Java schon fast vergessenen) Dingen wie dem Allozieren und Freigeben von Speicher sowie dem Suchen nach Memory Leaks beschäftigen und wild mit Pointern um mich schiessen. Als Entwicklungsumgebung für C dient in der AdNovum übrigens XEmacs. Das bedeutet doch eine Umstellung für Eclipse verwöhnte Programmierer, aber man gewöhnt sich recht schnell daran. Die Arbeit erleichtert haben die C Base Components. Das ist eine Bibliothek, die unter anderem einen sehr gut ausgebauten Logger und einen Konfigurationsmanager, mit dem man Informationen aus Konfigurationsfiles einlesen und auch manipulieren kann, bietet. Für die Kommunikation zwischen Runner und Agenten benutzte ich TCP, was mir die Gelegenheit gab, mich mit Netzwerkprogrammierung auseinander zu setzen und ein eigenes Protokoll zu implementieren. Für das Starten der Agenten auf anderen Rechnern nutzte ich RSH und SSH.

Damit man nicht nach jedem Testlauf mühsam alle Logfiles lesen muss, habe ich ein Perl-Skript geschrieben, das die automatisch gesammelten Logfiles parst und eine übersichtliche HTML Seite generiert. So konnte ich während des Praktikums Perl lernen und mich mit HTML beschäftigen.

Um die Infrastruktur, die auf den Testrechnern vorausgesetzt werden muss, möglichst gering zu halten, implementierte ich eine Funktion zum Verteilen der benötigten Dateien auf die Zielrechner und zum Einrichten der benötigten Umgebung (Dienste starten, etc.).

Die einzige Voraussetzung, die ein Rechner erfüllt haben muss, um vom DTF als Testrechner verwendet zu werden, ist die Zugriffsmöglichkeit über RSH oder SSH. Das DTF ist also in der Lage, sich selbst zu verteilen.

So ist es jetzt möglich, in den Nightly- und Weekly-Builds des Projektes auch verteilte Client-Server Tests laufen zu lassen, ohne die sonst üblichen Probleme, wie das vorgängige Verteilen der nötigen Files, das manuelle Starten der benötigten Services, oder der Unterhalt von Konfigurationsfiles auf allen Testrechnern.

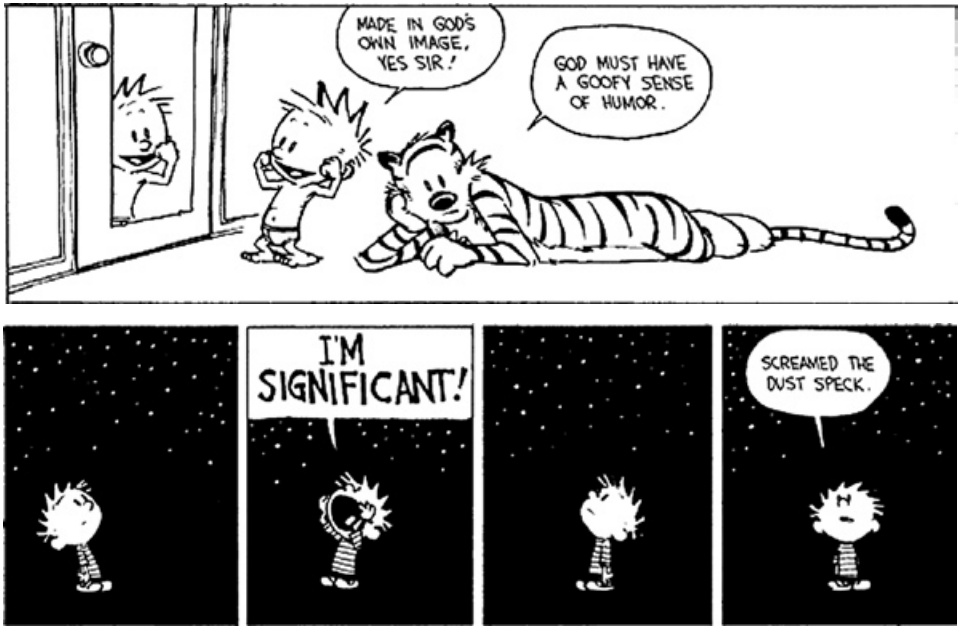
Am Schluss des Praktikums veranstaltet man einen „Tech-Znüni" zu dem alle Mitarbeiter eingeladen sind. Anders als der Name es vermuten lässt, gibt es dort allerdings in der Regel nichts zu essen. Man präsentiert die Ergebnisse seines Praktikums und diskutiert sie mit den übrigen Entwicklern.

Das Fazit

Mein Praktikum bei der AdNovum war sehr spannend und lehrreich. Ich hatte die Gelegenheit, neue Sprachen zu lernen und mich mit Netzwerkprogrammierung zu beschäftigen. Es war interessant zu sehen, wie das Arbeiten in einem grösseren Projekt mit vielen Beteiligten abläuft. Beim Lösen meiner Aufgabe war ich sehr frei; bei Problemen fand sich aber immer ein kompetenter Ansprechpartner. Abschliessend kann ich sagen, dass mir das Praktikum sehr viel Spass gemacht hat und ich die AdNovum als Praktikumsbetrieb sehr empfehle.

Links

[1] www.adnovum.ch
[2] www.nevis-web.com



TechTeam

Unix-Kurs

Teil 1 - Alltägliches Arbeiten Nicht zu vergleichen mit der Windows Kommandozeile

MATHIAS PAYER (GANNIMO@VIS ODER PAYERM@STUDENT)

An der ETH hat man ja Gelegenheit neben den ganzen Windows-Maschinen auch eine Unix- oder Linux-Maschine zu nutzen. Hier eine kleine Einführung dazu.

Der Vorteil der Kommandozeile (der Shell oder des Terminals) gegenüber einer Desktopoberfläche ist, dass man mit kurzen, leicht zu lernenden Befehlen eine Aufgabe viel schneller erledigen kann, als wenn man sich durch irgendwelche Menüs durchklickt und irgendwelche Icons quer über den Desktop zieht. Ein weiterer, für mich sehr wichtiger, Vorteil ist, dass man genau weiss, was das Kommando bewirkt und wie das Resultat aussehen sollte.

Ich könnte hier noch einige Seiten mit Vorteilen aufzählen, aber schliesslich wollt ihr ja etwas über Unix-Systeme lernen und diese benutzen können. Dazu werde ich euch noch kurz sagen, für was dieser Artikel gedacht ist und für was nicht. Mit dieser Serie möchte ich einen Einblick in die Unix Welt geben und wie man mit den dort vorhandenen Programmen effizient arbeiten kann. Auf was ich nicht genau eingehen werde, was aber in einem anderen Artikel kommen könnte ist die Installation eines Unix/Linux-Systemes.

Dateisystem

Das Unix Dateisystem unterscheidet sich stark vom Windows-Dateisystem. Die Unix-Mentalität besagt, dass alles eine Datei ist. Alles wird über Dateien angesprochen, sei das nun die Tastatur, die Grafikkarte, eine Harddisk, ein Terminal oder eine reguläre Datei. Die Wurzel des Unix-Dateisystems ist der Slash («/»), auf englisch Root. Unterhalb dieser Wurzel sind alle anderen Partitionen und Geräte eingehängt. Mit dem «mount»-Kommando kann man anzeigen, was für Partitionen und welche Dateisysteme momentan wo eingehängt («gemounted») sind.

Unter Windows wird jeder Partition einer Platte ein Buchstabe zugewiesen, unter dem auf die Partition zugegriffen werden kann. Unter Unix können die unterschiedlichen Partitionen irgendwo unterhalb der Wurzel eingehängt werden. Eine typische Ausgabe von mount ist:

```
> mount
/dev/hdb1 on / type ext3 (rw,errors=remount-ro)
/dev/mapper/md0-aes on /mnt/raid type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
```

Hier sieht man nun, dass die erste Partition auf der zweiten Harddisk als Wurzel eingehängt wurde. Danach wurde eine weiter Datenpartition unter /mnt/raid eingehängt. Die restlichen Einträge erlauben den Zugriff auf den Betriebssystemkern (Kernel) über virtuelle Dateien.

Der Unix-Kernel bietet eine Abstraktion der vorhandenen Hardware und zeigt gefundene Geräte unterhalb von /proc oder bei neuen Linux-Kernel auch unterhalb von /sys an. Falls man mit den dort vorhandenen Dateien arbeitet leitet der Betriebssystemkern diese Ein- und Ausgaben direkt an die dazugehörige Hardware weiter.

Auf einem Unix-System hat jede Datei (und jedes Verzeichnis) einen Besitzer und eine Gruppe zu der sie gehören. Nur der Besitzer und root, der Administrator, können den Besitzer einer Datei und deren Rechte ändern. Ausserdem hat jede Datei drei Felder für Berechtigungen. Eines für den Besitzer, eines für die Gruppe und eines für alle anderen Benutzer. In jedem Berechtigungsfeld steht, was mit dieser Datei gemacht werden darf. Es gibt 3 verschiedene Berechtigungen, lesen, schreiben und ausführen. Im Gegensatz zu Windows-Systemen, wird nicht über die Dateiendung bestimmt, ob eine Datei ausführbar ist, sondern über ihre Attribute. Sobald das Execute-Bit gesetzt ist, kann eine Datei ausgeführt werden. Siehe dazu auch die Erklärungen unter «man chmod».

In der Beispielsausgabe von `ls -lsa` sieht man, dass die Datei 0410-unix1.html dem Besitzer gannimo und der Gruppe users gehört. Der Besitzer darf die Datei lesen (read) und schreiben (write). Die Gruppe und alle anderen dürfen die Datei lesen. Zusätzlich werden die Grösse der Datei und das Datum der letzten Änderung angezeigt.

Beim zweiten Beispiel gehört die Datei dem Benutzer root und der Gruppe root, der Benutzer darf die Datei lesen, schreiben und als Programm ausführen. Alle Anderen dürfen die Datei lesen und ausführen.

Erster Kontakt mit einem Terminal

Nachdem ihr euch erfolgreich auf einer Linux oder Unix Maschine eingeloggt habt, könnt ihr ein Terminal starten. Je nach Oberfläche oder Konfiguration findet ihr das Konsolenprogramm im Menü oder direkt auf einer Schnellstartliste. Es gibt sehr viele verschiedene Terminals die ihr nutzen könnt, aber schaut für den Anfang mal nach einem xterm oder rxvt.

Sobald ihr ein Terminal geöffnet habt, seht ihr eine Kommandozeile. Das Terminal stellt einen Bildschirm zum Rechnersystem dar. Früher waren diese Terminals in unterschiedlichen Büros untergebracht und die Rechner waren noch «etwas» teurer.

Das Programm, das die Kommandozeile bereitstellt wird Shell genannt. Auch dort gibt es wieder sehr viele unterschiedliche Arten, da ist sicher für jeden Geschmack etwas dabei. In dieser Einführung werde ich versuchen so generell wie möglich zu bleiben und die vorgestellten Kommandos sollten auf allen möglichen Systemkombinationen funktionieren. Spätere Artikel werden je nachdem

```
> ls -lsa
8   -rw-r--r-- 1 gannimo users 5150   Oct 18 14:50 0410-unix1.html
504 -rwxr-xr-x 1 root    root  511400 Apr  8  2002 bash
```

einige Eigenschaften einer spezieller Shell genauer betrachten.

Nachdem eine Shell gestartet wurde befindet ihr euch im Normalfall in eurem Homeverzeichnis. Das ist der Ort wo ihr eure persönlichen Dateien, Ordner, Konfigurationen usw. speichert.

Beim Start einer Instanz eurer Shell werden normalerweise auch einige Init-Skripts aus eurem Homeverzeichnis abgearbeitet, mit denen Aliasse (Abkürzungen und Vereinfachungen für oft gebrauchte Befehle), das Aussehen des Prompts und andere Dinge eingestellt werden.

Das Hilfesystem

Eines der wichtigsten Kommandos auf Unix-Systemen ist «man» (kurz für Manual). Mit «man Kommandoname» kann man sich eine ausführliche Hilfe über das entsprechende Kommando anzeigen lassen, so liefert z.B. «man man» Erklärungen über das Manual-System. Für Fortgeschrittene hält das Man-System auch Erklärungen über Konfigurationsdateien bereit. Bei den vielen Kommandos, die man mit der Zeit in der Unix Welt kennenlernt, kann man nie alle Optionen und Einstellungen kennen. Da ist man oft dankbar über eine kurze Auflistung der wichtigsten Schalter. Diese findet man auch meistens in den Manuals und kann schnell bis zum relevanten Teil blättern.

Eine andere Möglichkeit, Informationen über ein bestimmtes Kommando anzuzeigen ist das Info-System. Mittels «info Kommandoname» kann man die sehr ausführliche und strukturierte Hilfe für ein spezifisches Kommando anzeigen lassen. Mit «info info» erfährt man, wie man das Info-System benutzen kann. Im Gegesatz zu einer Man-Seite, die monolithisch aufgebaut ist (alles auf einer lange Seite) ist das Info-System hierarchisch aufgebaut. Ein Info-Dokument ähnelt einer HTML-Datei mit Links zu anderen Seiten.

Falls man nur kurz die möglichen Schalter eines Kommandos erfahren will, reicht oft «Kommando -h», «Kommando -?» oder «Kommando --help». An diesen drei Möglichkeiten sieht man sehr gut wie Optionen an Programme übergeben werden können. So stellt z.B. «-h» eine Abkürzung für das ausgeschriebene Kommando «--help» dar. Kurze Optionen haben meistens einen Trennstrich und ausgeschriebenen Optionen einen doppelten Trennstrich.

Wichtige Kommandos

Nun folgen einige wichtige Kommandos, die ihr für den täglichen Umgang brauchen werdet. Falls euch die hier angegebenen Informationen nicht genügen, lasst euch die Hilfe oder die Manpage des Kommandos anzeigen. Nachdem ihr einige dieser Seiten gelesen habt werdet ihr euch auch an den Stil gewöhnen und könnt dann viel besser und auch schneller in diesen Handbüchern navigieren.

man Kurz für «manual». Dies ist das Unix-Handbuchsystem

info Unix-Informationssystem über Kommandos

w Kurz für «who». Zeigt die Anzahl eingeloggter Benutzer und deren letztes ausgeführtes Kommando an.

pwd Kurz für «Print (current) Working Directory». Gibt das aktuelle Verzeichnis aus.

ls Kurz für «list». Zeigt die vorhandenen Dateien in einem Verzeichnis an. Mit **ls -shal** werden die Grösse (Size) in für Menschen lesbarer (Human readable - Grösse in B, kB oder MB, je nach Grösse der Datei) in einem langen Format angezeigt.

mv Kurz für «move». Verschiebt Dateien oder Verzeichnisse. Falls der letzte Parameter ein Verzeichnis ist, können beliebig viele

Emotional Engineering –
unser Erfolgsrezept für die Zukunft der Computertechnologie

Kreative Hard- und Software

Du willst

- eines der weltweit schnellsten Computernetzwerke konstruieren oder einen berührungslosen Kartoffelsortierer bauen oder die Grundlagen für ein neues TV-Studio legen
- zusammen mit jungen, cleveren und kompetenten Leuten vielfältigste Projekte bearbeiten
- sowohl als Professional als auch als Mensch gefördert und gefordert werden

Du bist

- eine Fachfrau/ein Fachmann in Informatik, Elektronik, Physik oder Mathematik mit Fachhochschul-, Hochschulabschluss oder doktoriert
- engagiert und offen für Neues
- team- und lernfähig

Wir

- sind ein unkonventionelles Hightech-Unternehmen
- entwickeln anspruchsvolle Produkte (Hardware Design bis GHz, Software von Assembler bis OO)
- gehen neue Wege
- denken quer

Supercomputing Systems
Technoparkstrasse 1 · 8005 Zürich
Tel.: 01/445 16 00 · Fax: 01/445 16 10
E-Mail: sekretariat@scs.ch · WWW: http: // www.scs.ch

POLY
BUCHHANDLUNG

Unsere Rabatte für Studierende

20 % auf vdf-Publikationen
20 % auf Bücher von Prof. A. Seiler
10 % auf nicht preisgebundene Bücher

Bei Sammelbestellung & Sammelabholung:
5 % ab 10 Exemplare
10 % ab 20 Exemplare
12.5 % ab 50 Exemplare
15 % ab 100 Exemplare

Öffnungszeiten: Mo - Do: 10:00 - 16:30 Uhr
Fr: 10:00 - 15:30 Uhr
in den Ferien: Di, Mi, Do: 10:45 - 15:15 Uhr

ETH Hönggerberg
HPI E16.1
8093 Zürich
Tel: 01 633 27 78
www.books.ethz.ch

Öffnungszeiten: Mo - Do: 09:30 - 16:30 Uhr
Fr: 09:30 - 15:30 Uhr
in den Ferien: Mo - Fr: 11:00 - 15:00 Uhr

ETH Zentrum
MM B96
8092 Zürich
Tel: 01 632 42 89
www.books.ethz.ch

Dateien und Ordner davor genannt werden, die alle verschoben werden.

cp Kurz für «copy». Kopiert Dateien oder Verzeichnisse. Falls der letzte Parameter ein Verzeichnis ist, können beliebig viele Dateien davor genannt werden, die alle kopiert werden.

rm Kurz für «remove». Löscht Dateien und Verzeichnisse

rmdir Kurz für «remove directory». Löscht ein (leeres) Verzeichnis.

ln Kurz für «link». Mit diesem Kommando kann man eine Verknüpfung erstellen. Für eine symbolische Verknüpfung lautet der befehlt wie folgt: **ln -s** ***Datei Verknüpfung***.

chmod Mit diesem Kommando können die Berechtigungen für eine Datei oder ein Verzeichnis geändert werden.

chgrp Damit kann die Gruppe, der die Datei oder das Verzeichnis gehört, geändert werden.

chown Hiermit kann der Besitzer, dem die Datei oder das Verzeichnis gehört, geändert werden.

du Kurz für «disk usage». Dieses Kommando zeigt den Platzverbrauch aller Dateien in den Unterverzeichnissen an. Der Parameter «-h» gibt sinnvolle Grössenangaben aus.

df Kurz für «disk free». Zeigt pro eingehängte Partition den freien und verwendeten Speicherplatz an.

tar Kurz für «tape archiver». Das ist die Unix-Variante von ZIP-Dateien. Nur sind Tar-Dateien nicht komprimiert, dazu muss ein externes Kommando wie gzip verwendet werden. Neuere Tar-Versionen können diese externen Programm über Parameter aufrufen. **tar cvzf** ***archiv.tar.gz Datei1 Verzeichnis2 ...*** erstellt ein Archiv mit den angegebenen Dateien, **tar zxvf** ***archiv.tar.gz*** entpackt das gegebenen Archiv.

gzip Ein Komprimierungstool, das einzelne Dateien komprimiert.

gcc Der Gnu C-Compiler, wird für viele freien Software-Projekte verwendet.

make Das Unix-Make Tool wird gerne genutzt, um grössere Projekte zu kompilieren und installieren.

nano/pico Zwei Editoren, die vor allem leicht für den Einsteiger sind. Beide sind eigentlich selbsterklärend.

ps Kurz für «process status». Zeigt die laufenden Prozesse auf einem System an. **ps auxwf** gibt eine baumartige Übersicht über alle auf einem System laufenden Prozesse.

top Zeigt die Prozesse an, welche momentan am meisten Systemleistung brauchen und aktualisiert sich alle 10 Sekunden.

kill Mit **kill** ***pid*** kann man einen Prozess, der sich z.B. nicht mehr durch Drücken von Ctrl-C beenden lässt, abschiessen. Mit **kill -9** ***pid*** wird der Prozess sofort getötet und beendet. Die pid-Nummer erfährt man über das ps-Kommando.

VisAktiv

Ich wollte nie mit Datenbanken arbeiten

RES - INSGEHEIM EIN DATENBANK-FAN

Da bin ich nun schon ein Jahr nicht mehr beim VIS und habe immer noch keinen Abschiedsartikel geschrieben. Ich sollte mich schämen. Und einen Praktikumsbericht habe ich auch nie abgeliefert.

Für all diejenigen, die mit meinem Namen nichts anfangen können, hier erst mal einen kurzen Überblick über meine VIS-Tätigkeit.

Als ich etwa 2001 in den VIS-Vorstand eintrat (am Anfang meines Fachstudiums), übernahm ich erst mal das Amt des Quästors, das ich bis fast zum Ende auch behielt. Die Zeit war nicht unbedingt die Beste, Informatiker wurden mehr entlassen als eingestellt und so brach der Umsatz der Kontaktparty (bekanntlich Haupteinnahmequelle des VIS) stark ein. Mein Wirken war nicht sonderlich erfolgreich, erst nach rund zwei Jahren unter Alex Präsidialzeit wurden die Budgets wieder ausgeglichener und unterdessen soll ja auch die KP wieder etwas erfolgreicher sein.

Daneben angelte ich mir etwas später noch den Job als Aktuar, mit dem Ziel Protokolle zu schreiben, die nachfolgende Generationen zum Schmunzeln bringen (ob mir das gelang, müsst ihr Beat fragen). Schliesslich engagierte ich mich in der Unterrichtskommission und in der Studienreformkommission (SRK).

Langes Rumsitzen

Die SRK sollte das Bachellor/Master-Studienreglement ausarbeiten. Das bedeutete für mich ein Jahr lang jeden Freitag morgen mit durchschnittlich fünf Professoren und einem Doktoranden zusammen zu sitzen. Ich weiss, ich habe einigen Kollegen mal in einer wütenden Stunde versprochen, einen Artikel über diese Sitzungen zu schreiben, aber unterdessen bin ich darüber hinweg und sehe es als interessante Erfahrung. Ich hoffe nur, dass meine fehlenden diplomatischen/verhandlungstechnischen-Talente den Bachellor-Studis nicht allzu grosse Probleme machen. Aber nach meinem Abgang wurde auf dem Gebiet auch noch intensivst weiter verhandelt und ich gebe zu, ich weiss mittlerweile gar nicht mehr, wie das Reglement nun aussieht.

Deutschlandreisen

Nebenbei versuchte ich noch etwas mehr Kontakt zu anderen Unis zu schaffen, so nahm ich zweimal

an einer KIF (=Konferenz deutschsprachiger Informatik Fachschaften) teil. Ein sehr unterhaltsamer Anlass, bei dem man fünf Tage lang viel diskutiert (über das Studium, aber auch über alle möglichen anderen Themen), spielt (aber nur Brettspiele), grüne Katzen näht, Spass hat aber generell zu wenig schläft. Ich hoffe, dass der VIS-Vorstand irgendwann in Zukunft sich dazu entscheiden kann, wieder einmal an einer KIF teil zu nehmen. Die nächste findet übrigens im November in Jena statt und die übernächste irgendwann im Frühling in Wien.

Studium, ...

Meine Fächer Auswahl im Fachstudium konsentrierte sich stark auf Wissenschaftliches Rechnen und auf die «Maschinen Lernen»-Fächer. Die systemnahen Fächer beherrschte ich leider nie, so fiel ich sowohl in Systemprogrammierung wie auch in Systemsoftware durch, und Datenbanken interessierten mich einfach nie.

... Praktikum ...

Mit Hardware habe ich immer noch nichts zu tun, aber bei den Datenbanken lag ich wohl ziemlich daneben. Schon mein Praktikum bei «Ernst Basler + Partner» war auf dem Gebiet der Geoinformation Systems, ich arbeitete dort an einer Software mit, die einer Rückversicherung ermöglichen sollte, ihr Risikoportfolio nach gewissen Schadensszenarien (Hurrikane in Florida oder Erdbeben in Europa als Beispiele) zu beurteilen. Ein sehr spannendes Projekt, bei dem ich für meine Verhältnisse ziemlich viel mit Datenbanken arbeitete. Leider brauchte die Firma am Ende meines Studiums in der Informatik keine Mitarbeiter und so machte ich mich auf die Suche nach einem anderen Arbeitgeber.

... und Arbeit

So landete ich am Schluss bei einem Lebensversicherer (der Titel dieses Artikels sagt alles). Und wie nicht anders zu erwarten arbeite ich wiederum schwergewichtig mit Datenbanken. Die eingesetzte Software ist topmodern, äh, naja fast...

Ich arbeite bei dem System mit, in dem die Verträge der Einzelversicherung (im Unterschied zur Kollektivversicherung, den Pensionskassen) verwaltet werden. Unser System ist in C geschrieben (C, nicht C++!) und ist seit 1996 im Einsatz. Dies ist jedoch das momentan neuste System, in das gerade fleissig Altbestände migriert werden. Die Altbestände sind im Moment noch (unter anderem) auf einem Bull-Rechner, ca. von 1975, und laufen auf einem Cobol-System. Dass migriert werden muss hat natürlich nichts mit dem Alter des Systems zu tun, sondern ist hauptsächlich Bull zu verdanken, denn die Maschinen werden nur noch bis 2008 unterstützt...

Und so arbeite ich heute an einem System, dass einen recht grossen Datenbestand in einer Oracle-Datenbank verwaltet, und setze fast täglich ein paar SQL-Statements direkt auf die DB ab...

Das soll reichen

Ich hoffe ihr habt alle noch viel Spass während eures Studiums (zumindest diejenigen Leser, die noch studieren) und viel Erfolg bei allfälligen Prüfungen. - Wenigstens den doppelt kompensierten Schnitt konnten wir euch am Schluss mit Mühe und Not (und viel Schützenhilfe vom Rektor) noch ersparen.

Mir bleiben vom VIS nun noch ein Raclette-Ofen, viele gute Erinnerungen und die Hoffnung noch an viele VISKAS kommen zu dürfen.

Auch Studenten können ihr Wissen anwenden

ETHJUNIORS - VERMITTELN ARBEIT

Herzlich Willkommen an der ETH heisst euch auch ETH juniors. Willkommen an einer der renommiertesten Hochschule weltweit. Durch diese Hallen wandelten schon so viele berühmte und kluge Persönlichkeiten, dass einem fast Angst und Bange wird. Neue Publikationen, Erfindungen und wissenschaftliche Errungenschaften gehören an der ETH zur Tagesordnung.

Doch die ETH ist nicht nur Forschungsplatz. Sie ist auch Ausbildungsstätte. So müssen auch alle die Studenten, die vielleicht weniger an einer wissenschaftlichen Karriere interessiert sind, die genau gleichen Vorlesungen durchlaufen wie alle die zukünftigen Einsteins und Newtons. Sie pauken stundenlang Theorie und fragen sich, wozu sie Sachen lernen wie zum Beispiel eine Kovarianzmatrize im Kopf auszurechnen.

Eine Antwort auf diese Frage können wir Dir auch nicht geben. Aber vielleicht können wir Dich motivieren die Strapazen durchzustehen. ETH juniors ist die «Junior Entreprise» der ETH Zürich. Wir versuchen eine Brücke zwischen der Wirtschaft und der ETH zu schlagen. Firmen können etwelche Projekte an uns auslagern, die ein spezifisches Wissen erfordern das an der ETH gelehrt wird. Du kannst mit gut bezahlten Projektarbeiten dein Taschengeld aufbessern und erkennst, welches Wissen auch ausserhalb der ETH zählt.

Geh auf unsere Homepage [1] und trage dich in die Datenbank ein. Du wirst dann immer wenn ein neues Projekt ansteht automatisch per e-mail angeschrieben. Traue dich und nutzte die Chance praktische Ehrfahrung zu sammeln.

ETH juniors organisiert aber auch viele Events. Immer in Zusammenarbeit mit Firmen aus allen möglichen Branchen. Die Firmen wollen euch kennenlernen. Und ihr sollt die Firmen kennenlernen. Es tut immer wieder gut zu merken, dass man als ETH Absolvent begehrt ist in der Wirtschaft. So kann man sich viel leichter motivieren für all sie Prüfungen die man schreiben muss. Es lohnt sich.

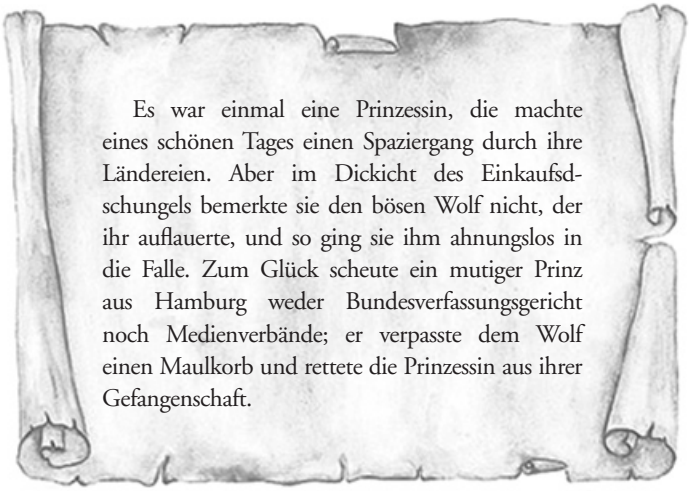
Links:

[1] www.juniors.ethz.ch

Alles was Recht ist

Über Prinzen, Prinzessinnen und das Märchen der Privatsphäre

DANIEL MARKWALDER - MÄRCHENERZÄHLER



Es war einmal eine Prinzessin, die machte eines schönen Tages einen Spaziergang durch ihre Ländereien. Aber im Dickicht des Einkaufsdschungels bemerkte sie den bösen Wolf nicht, der ihr auflauerte, und so ging sie ihm ahnungslos in die Falle. Zum Glück scheute ein mutiger Prinz aus Hamburg weder Bundesverfassungsgericht noch Medienverbände; er verpasste dem Wolf einen Maulkorb und rettete die Prinzessin aus ihrer Gefangenschaft.

Was interessiert das einen Informatiker?

Kann man sich mit Recht fragen... Vielleicht tatsächlich nicht sehr viel? Ein Blick in die Kristallkugel - genannt inforum - lässt aber doch vermuten, dass sich Informatiker durchaus für Persönlichkeitsschutz interessieren. Und offen gesagt auch interessieren sollten, schliesslich eröffnet gerade die Informatik allen grossen Brüdern dieser Welt ganz neue Möglichkeiten: Unbeschränkte Speichermöglichkeiten (mengen- und zeitmässig) und unkontrollierbare Verknüpfungsmöglichkeiten sind nur zwei der Möglichkeiten, welche das Datensammeln in einen ganz neuen Kontext stellt. In diesem Artikel geht es allerdings weniger um den datenschutzrechtlichen Aspekt der Privatsphäre, sondern aus aktuellem Anlass mehr um die Zulässigkeit medialer Berichterstattung.

Caroline-Urteil

Prinzessin Caroline von Monaco war beim Einkaufen, Sport und Diner mit Freunden „unautorisiert“ fotografiert worden. Das von ihr geforderte Verbreitungsverbot der Bilder war durch deutsche Gerichte mit der Begründung abgeschmettert,

Caroline sei so berühmt, dass ihre Privatsphäre mit dem Übertreten der Hausschwelle ende. Solche Berühmtheiten dürften nur dann nicht fotografiert werden, wenn sie sich für die Öffentlichkeit erkennbar an einen „abgeschiedenen Ort" zurückgezogen hätten. Carolines Promi-Anwalt Matthias Prinz (der heisst tatsächlich so...) hat aber nach elfjährigem Rechtsstreit (!) vor dem Europäischen Gerichtshof recht bekommen. Gemäss diesem Urteil wird auch Prominenten eine Privatsphäre in der Halböffentlichkeit garantiert. Für eine Berichterstattung ist folglich immer die Zustimmung erforderlich, ausser wenn dadurch Informationen oder Ideen verbreitet werden, welche zu einem Diskurs im Allgemeininteresse der Gesellschaft dienen.

Dieses sogenannte „Caroline-Urteil" ist - von der deutschen Regierung unangefochten - seit dem 24. September 2004 rechtskräftig. Ein Urteil, welches gemäss Interpretation gewisser Zeitungen, den investigativen Journalismus verunmöglichen wird...

Zensur

Soll man folglich nicht mehr darüber berichten dürfen, wenn ein Fotomodell und Tierschutzaktivistin im Nerzmantel dinieren geht? Oder was ist mit dem Politiker, der sich für weniger Alkohol im Blut stark macht und beim Fahren im Suff erwischt wird? Solche Dinge wollen wir selbstverständlich erfahren! Aber wie ist das mit den Paparazzis, die hinter Lady Diana herjagten und sie ins buchstäbliche Unglück trieben?

Wie eigentlich immer im Recht, geht es auch hier um ein Abwägen zwischen verschiedenen Interessen: Dem Interesse der Person auf Privatsphäre einerseits und andererseits dem öffentlichen Interesse an (öffentlichen) Personen.

Nun aber konkret

Die Rechtslage rund um Berichterstattung und Persönlichkeitsschutz ist in der Schweiz in Artikel 28 ZGB verankert:

Art. 28 ZGB:

1 Wer in seiner Persönlichkeit widerrechtlich verletzt wird, kann zu seinem Schutz gegen jeden, der an der Verletzung mitwirkt, das Gericht anrufen.

2 Eine Verletzung ist widerrechtlich, wenn sie nicht durch Einwilligung des Verletzten, durch ein überwiegendes privates oder öffentliches Interesse oder durch Gesetz gerechtfertigt ist.

Wie stark der Eingriff sein darf, bis er als widerrechtlich einzustufen ist, hängt davon ab, ob man Politiker, Promi oder Normalbürger ist. Auch die Schwere des Eingriffs kann man in drei Kategorien einteilen, nämlich in Intimsphäre (die eigenen vier Wände), Privatsphäre (Kaufhaus, Arbeitsplatz, etc.) und öffentlicher Bereich (Demo, Fernsehstudio, etc.). Dadurch kann man folgende 3x3-Matrix für die Zulässigkeit medialer Berichterstattung aufstellen:

	Intimsphäre	Privatspäre	Öffentlicher Bereich
Politiker	*Problematisch*	*Grds. erlaubt*	*erlaubt*
Prominente	*Nicht erlaubt*	*Grds. nicht erlaubt*	*erlaubt*
Normalbürger	*Nicht erlaubt*	*Nicht erlaubt*	*Grds. erlaubt*

Das bedeutet...

Der Normalbürger soll also nicht beim Grillieren auf der IFW-Terasse fotografiert werden dürfen. Wer jedoch bei einer Demo eine Rede hält, muss sich die Berichterstattung auch ohne seine Einwilligung gefallen lassen. Heikel und rechtlich umstritten ist das Zwischending: Darf der Demoteilnehmer fotografiert und identifiziert werden oder hat er das Recht, dies zu verhindern (Stichwort Vermummungsverbot)? Gegen das Fotografieren spricht die freie Meinungsäusserung (vielleicht droht mir ja die Kündigung, wenn der Arbeitgeber mich auf der Strasse sieht...). Hingegen wird natürlich durch vermummte Demonstranten erfahrungsgemäss viel mehr Schaden angerichtet. Das öffentliche Interesse an Ruhe und Ordnung ist dabei wohl akuter bedroht als eine potentielle Kündigung, allerdings trifft es denjenigen, der den Job verliert viel unmittelbarer als denjenigen, der im schlimmsten Fall eine Scheibe ersetzen muss. Man sieht, dass die Abwägung nicht ganz einfach zu ziehen ist und dass sie (auch) von der politischen Einstellung abhängt....

Volenti non fit iniuria

Dem Einwilligenden geschieht kein Unrecht, dies ist die erste wichtige Ausnahme der obigen Tabelle: Auch die intimsten Details dürfen natürlich öffentlich gemacht werden, wenn der Betreffende dem zustimmt (z.B. Big Brother-Serie). Die zweite Ausnahme ist das bereits oben diskutierte öffentliche Interesse, welches gegen die Interessen des Betroffenen abgewogen werden muss.

Beim besoffenen Politiker überwiegt das öffentliche Interesse natürlich allemal und ziemlich sicher auch bei der Nerzmantelträgerin. Insofern ist das laute Geheule der Regenbogenpresse über Zensur und Maulkorb ein bisschen übertrieben, wenn nicht gar ein Märchen.

Wenn das Geld im Kasten klingt

Im Falle der Privatsphäre der Prominenten hat das Caroline-Urteil einen Systemwechsel gebracht: Wo früher die Berichterstattung grundsätzlich zulässig und ausnahmsweise verboten war, ist sie heute grundsätzlich verboten und ausnahmsweise erlaubt. Die Privatsphäre (auch) der Promis verstärkt zu schützen ist durchaus eine positive Entwicklung. Dabei ist es aber mindestens interessant, dass Politiker schwächer geschützt werden als Prominente! Dies obwohl Caroline, Madonna, Britney und Co. von der Presse leben und ihr Leben so öffentlich wie nur irgendwie möglich machen. Und weil sich dies mit Sicherheit nicht ändern wird, wird der Paradigmenwechsel auch nicht zur Folge haben, dass wir weniger Privates oder Intimes über Prinzen und Prinzessinnen zu hören und sehen bekommen, vielmehr wird die Privatsphäre vermehrt zum rechtlich geschützten und damit lizenzierbaren und kommerzionalisierbaren Gut. Damit wird eigentlich nur ein Trend bestätigt, der schon lange stattgefunden hat: Privatsphäre verkaufen ist spätestens seit Cumulus eine Alltäglichkeit! Womit man schliesslich zur ernüchternde Erkenntnis gelangt, dass es nicht die Paparazzis sind, die unsere Privatsphäre ernsthaft gefährden (auch nicht die Datensammler), sondern letztlich wir selber, weil sie uns viel zu wenig wert ist! Wir finden zwar alle, dass die Privatsphäre zu schützen ist, ergötzen uns aber an Big-Brother-Knatsch, haben unsere Cumuluskarte im Portemonnaie und finden Mailverschlüsselungen zu mühsam... Oder wie schon Winston Churchill treffend bemerkt hat: „*Wer immer wieder liest, wie schädlich das Rauchen ist, hört irgendwann auf - zu lesen.*“

Links

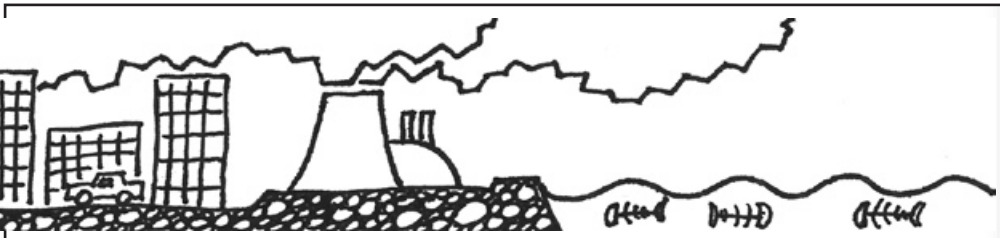
http://www.admin.ch/ch/d/sr/210/index.html
http://www.nzzamsonntag.ch/2004/09/24/em/page-article9VD1B.html
http://www.altenburger.ch/pdf/pub_immaterial01.pdf

Die Welt gemäss Beni Koller

MICHAEL GROSSNIKLAUS - WÄSCHT SEINE HÄNDE AUF DEM WC

Beni Koller genoss einen Drink auf der Feier eines Arbeitskollegen. Als er seinen Wodka mit Red Bull verdünnte, fiel ihm das Datum auf, das auf dem Boden der Büchse aufgedruckt war. Es wies das Getränk als für zwei weitere Jahre haltbar aus. Beni fragte sich, wie diese Daten zustande kamen. Wurde ein Verfalldatum empirisch im Experiment ermittelt, indem tausende von Dosen abgefüllt und aufbewahrt wurden, um dann jeden Tag eine zu öffnen und die Bekömmlichkeit des Getränkes zu testen? Oder gab es vielleicht Tabellen, die die Haltbarkeit von Inhaltstoffen angeben und aus denen die Gesamthaltbarkeit der Mixtur abgelesen werden konnte? Vielleicht wurde der Verfall auch chemisch über eine kurze Zeit ermittelt und dann mathematisch extrapoliert. Trotzdem, niemand den er kannte, konnte ihm erklären, wie es funktionierte und doch akzeptierten es alle und richteten sich danach.

Händetrockner, wie sie häufig in den Toiletten von Restaurants zu finden sind, stellen ein ähnliches Rätsel dar. Beni fragte sich, aus welchen Interessen diese installiert wurden. Sicher nicht zum Vorteil des Benutzers, denn ihm war es bislang nie gelungen, seine Hände damit trocken zu kriegen. Noch jedes Mal hatte er die Geduld verloren und am Ende Toilettenpapier oder seine Hosen benutzt. Vermutlich waren die Geräte umweltfreundlicher als Handtücher. Aber auch das konnte sich Beni nicht vorstellen, obwohl der stattliche grüne Baum auf den Gehäusen der Geräte eines amerikanischen Herstellers dies suggerierte. Vermutlich war der einzige, der von diesem Gerät profitierte, der Betreiber, der sich das Nachfüllen der Handtücher sparen konnte. Perfide dabei war, dass dem Benutzer durch die Installation eines modernen technischen Gerätes Kundenfreundlichkeit vorgetäuscht wurde, die nicht im Geringsten existierte! Das gleiche passierte in den Augen von Beni beim Siegeszug des Personal Computers.



Heute hatten Hinz und Kunz einen solchen Rechner zuhause, dessen Potential sie weder verstanden, noch ausnutzen konnten. Die Industrie hatte es in genialer Weise verstanden, ihr Unvermögen, ein nützliches Produkt für den Alltag zu entwickeln und die damit verbundenen Probleme, auf den Kunden abzuschieben. Auf dem Gipfel der Frechheit erlaubten sich Hersteller und selbsternannte „PC Doktoren" auch noch, diese Leute für Support zur Kasse zu bitten oder vertrauten darauf, dass sich in deren Bekanntschaft ein „Computergenie" befand, das die Probleme gegen eine Tafel Schokolade löste.

Beni beschlich das Gefühl, dass die Gesellschaft immer mehr in einer Scheinwelt lebte und nicht realisieren wollte, was um sie herum abging. Hatten einige wenige sich gegen die Menschheit verschworen? Er glaubte nicht, dass dies der Fall war, denn es schien ihm gar nicht nötig, sich gegen die Menschen zu verschwören. Viele hatten seiner Meinung nach das aktive Bedürfnis sich vor der Realität in eine Phantasie zu flüchten! Wie anders war es sonst möglich, dass die Milliarden von Science Fiction Fans mit ihren humanistischen Idealen und ihren Visionen einer perfekten Zukunft nie zur Stelle waren und protestierten, wenn gerade diese Ideen mit Füssen getreten wurden? Vermutlich sassen sie zuhause und sahen ihre Lieblingsserie am Fernsehen... Wie kam es, dass die Menschheit mit grossem Interesse nach ausserirdischem Leben suchte, wenn gleichzeitig die Mehrheit der Menschen den andersfarbigen Nachbar als Bedrohung ansah und nichts mit ihm zu tun haben wollte? Beni Koller schaute auf und sah, dass er alleine im Raum geblieben war. Er stand auf und machte sich auf die Suche nach seinen Kollegen. Er wollte schliesslich kein Einzelgänger sein.

Bilderrätsel

Nach dem Riesenerfolg des letztsemestrigen Rätsels, gibt es dieses Semester wieder ein neues. Löse das Bilderrätsel und schicke deine Lösung an videosessions@vis.ethz.ch.
Zu gewinnen gibt es ein Videosession-Erlebnispaket, einlösbar an einer Videosession deiner Wahl. Darin inbegriffen ist:
Ein Logenplatz, ein Vis-Tshirt, eine Vis-Tasse und eine freiwählbare Pizza.

Tipp

Sechs Bilder haben etwas mit Film, Fernsehen und Fiktion zu tun.

~~1-3~~

1

1

1 3 2=F

3

~~4-6~~

1

[7]+1

4 doppelt

GLANCE
A BOV COMPANY

Beratung heisst für uns: Zuhören, begreifen und den Kunden verstehen.

Auf dieser Basis finden unsere Spezialisten die optimale Lösung auch für Ihre Bedürfnisse.

IT Consulting

www.glance.ch

AZB
PP/Journal
CH - 8092 Zürich

Falls unzustellbar bitte zurück an:
Verein der Informatik Studierenden
RZ F17.1
ETH Zentrum
CH-8092 Zürich

Agenda

November

10. November Vis Erstsemestrigenfest
27. November Polyball

Dezember

6. Dezember Dr Samichlaus chunnt

VSETH
VERBAND DER STUDIERENDEN AN DER ETH