Visionen

12
Dezember
92

LO
GO

VIS Logo Wettbewerb
Reisebericht Dresden
Spielzeuge der Informatiker, 2

Visionen Herausgegeben vom Verein der Informatikstudierenden an der ETH Zürich

Adressen

Aktuarin: Grete Danielsen
Dohlenweg 26
8050 Zürich, Tel 01 / 302 48 97
e-mail: gcdaniel@iiic.ethz.ch

Exkursionen: Christian Franz
Sonneggstr. 61
8006 Zürich, Tel. 01/ 261 26 96
e-mail: cfranz@iiic.ethz.ch

Feste & Kultur: Frank Möhle
Dielsdorferstrasse 7
8155 Niederhasli, Tel. 01 / 851 03 21
e-mail: fmoehle@iiic.ethz.ch

Präsident: Florian Schlotke
Roswiesenstr. 161
8051 Zürich, Tel. 01/ 321 46 23
email: fschlotk@iiic.ethz.ch

Quästor: Daniel Kluge
Irringersteig 3
8006 Zürich, Tel. 01/ 252 04 14
e-mail: dankluge@iiic.ethz.ch

Redaktor: George Fankhauser
Schaffhauserstr. 298
8050 Zürich, Tel. 01/ 312 10 32
e-mail: gfankhau@iiic.ethz.ch

Verleger: Boris Nordenström
Hardstrasse 324
8005 Zürich, 01 / 273 24 80
e-mail: banordens@iiic.ethz.ch

Visinfo(Infosystem): Maxim Samo
Forchstrasse 245
8032 Zürich, Tel. 01/ 381 17 50
e-mail: samo@nessie.cs.id.ethz.ch

Vordiplome: Marcel Waldvogel
Hägetstalerstr. 37
8610 Uster, Tel. 01/ 941 61 94
e-mail: mwaldvog@iiic.ethz.ch

Impressum

Herausgeber:
Verein der Informatikstudierenden an der ETH Zürich.

Verleger: Boris Nordenström
Redaktor: George Fankhauser

Adresse Verlag & Redaktion:
VIS
Verein der Informatikstudierenden
Haldeneggsteig 4, IFW B29
ETH Zentrum
8092 Zürich

Tel: 01 254 72 12 (Mo-Fr, 1215-1300)
e-mail: vis@iiic.ethz.ch

Postscheckkonto 80-32779-3
Präsenzzeit: Mo..Fr: 1215..1300

Auflage: 1600
Inseratenpreis/Seite 500.-
Jahresabonnement 15.-

Redaktions- und Anzeigeschluss für die nächste Ausgabe:

29. Januar 1992

Visionen

Tschau Zame

Wie immer geht auch dieses Jahr, so kurz vor Weihnachten alles drunter und drüber. Ich halte mich deshalb bewusst und mit allen Konsequenzen kurz. Ich hoffe Ihr hattet alle einige besinnliche Festtage. Fürs neue Jahr wünsche ich allen die nötige Portion Glück...

PROSIT NEUJAHR!

-flo

Unterwegs mit dem VIS... Heute: Dresden

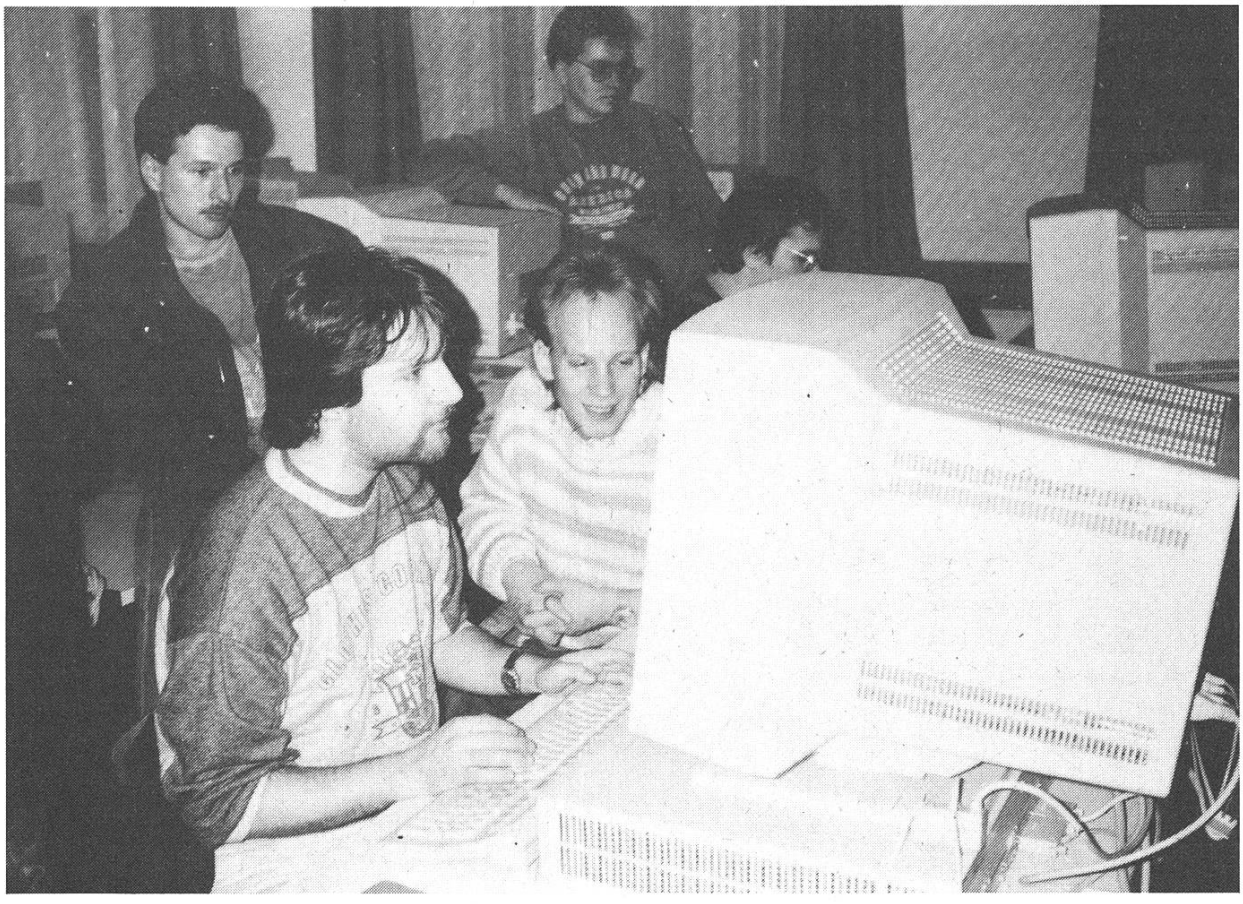
Was täten wir ohne Computer? Wir schreiben Mittwoch, den 3. Dezember 1992. Ein Informatikstudent sitzt gemütlich im Sun-Raum und denkt sich nichts böses. Plötzlich wird er angetalkt: "Ich geh jetzt zum Bahnhof, kommst du auch?", "Bahnhof wie – warum", "Na ja, du kommst doch mit nach Dresden", "Dresden? Ach ja morgen dann...", "Nein jetzt in 15 Minuten", "Aber... WAAAS?, oh sh.."

15 Minuten später sind wir dann doch vollzählig am HB und haben sogar noch einen Schlafsack und ein paar Fränkli für unseren Kurzentschlossenen organisieren können. So kam es, dass sich 8 Informatiker und eine Informatikerin aus Zürich auf den Weg machten, die neuen deutschen Lande zu erkunden. Auf der Fahrt nach Basel werden fleissig Stadtpläne und Reiseführer gescant. Basel, the point of no return. Der letzte Stop im zivilisierten Teil der Welt. Die Delegation des VIS deckt sich mit all dem ein, was man eben so braucht um eine Nacht im Liegewagen zu überleben.

Dank der fürsorglichen Hilfe des Schlafwagenschaffners erreichen wir

12 Stunden und einige Bier später Dresden Hbf. Die Freude ist gross, wir werden schon erwartet, um 5:30 Uhr am Morgen, man stelle sich das vor. Kaum haben wir uns versehen, ist unser Gepäck schon in einem Trabi verschwunden und Minuten später finden wir uns nach einer aufregenden Tramfahrt in einem der legendären Dresdner Studentenclubs wieder. Der angebotene Kaffee macht 8 munter und beschert dem neunten Magen/-Darmverstimmungen für die nächsten Tage.

Unser erster Besuch gilt der Informatikabteilung der TU-Dresden. Die meissten Unis in Ostdeutschland sind im Moment einem tiefgreifenden Wandel unterworfen. Die Strukturen werden geändert, der Lehrkörper erneuert und Geld hat sowieso niemand. Viele Studis sind desshalb ziemlich verunsichert. Doch die Informatikinfrastruktur kann sich durchaus sehen lassen. So wurde ein ganzer Computerraum von der Industrie gesponsort und mit DECStations eingerichtet. Marcel und Carlo können es nicht lassen und loggen sich gleich nach

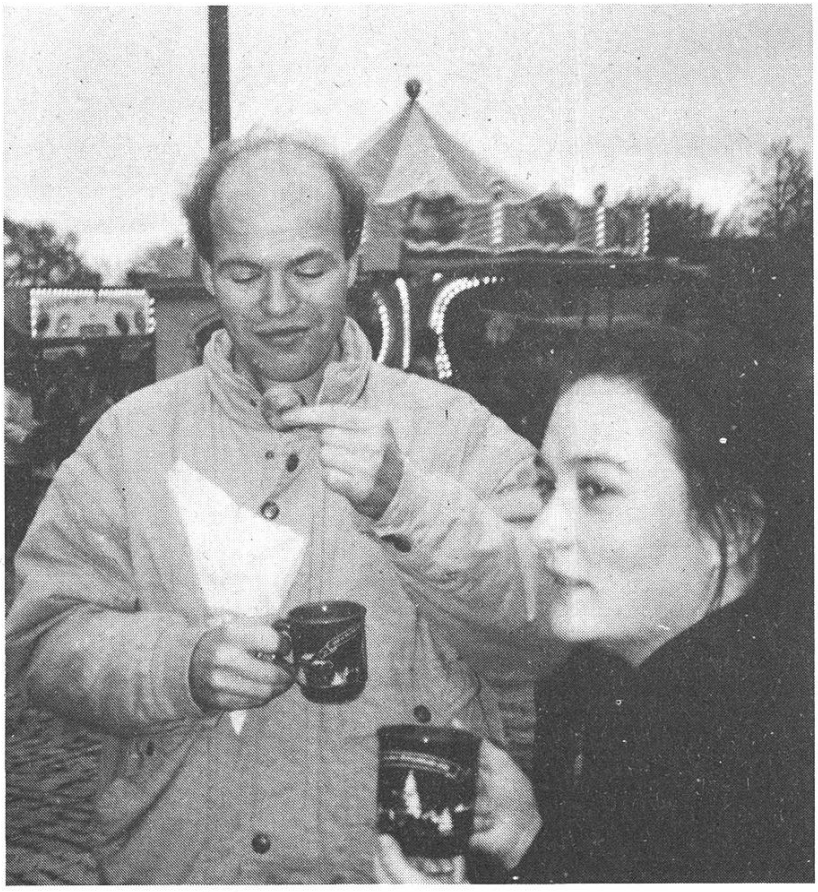


Zürich ein um via EZInfo einen zünftigen chat zu starten. (Bitte keine Fragen bezüglich der Performance und vorallem nicht nach dem Sinn der Übung). Ein PC und Mac-Raum wird entdeckt jedoch kein CERES-Raum. Diese Hinterwäldler...

Die folgende Vorlesung über Mensch-Maschine-Kommunikation ist ein besonderes Erlebnis, das ihr euch besser persönlich erzählen lasst. Jedenfalls haben wir schwer mit dem sich bemerkbar machenden Jetlag zu kämpfen. Wir stärken uns dann erst mal für sage und schreibe einemarkachzig in der Mensa.

Der Nachmittag beschert den einen ein kleines Nickerchen, die anderen wagen den ersten Schritt in die Stadt, kommen aber nicht weit über den grossen Dresdner Weihnachtsmarkt, den "Stritzelmarkt" hinaus. Man verzettelt sich in der Aufgabe: "Wie scanne ich am effizientesten einen Weihnachtsmarkt ohne eine Bude auszulassen oder doppelt zu besuchen und trotzdem den maximalen Genuss an Christstollen, Glühwein, Kräppelchen, Mandeln, Waffeln, Fischbrötchen etc. zu erreichen." Die Lösungen werden anschliessend im ersten Cafe am Platz diskutiert. Nachdem wir - vier, einen absolut strangen Dialekt sprechenden, Informatiker – einer Dresdner Verkäuferin auseinandersetzen konnten, wieviel Brot es für ein Fondue für 15 Personen braucht, traten wir erschöpft den Heimweg in den Kellerclub an. Dort fand dann ein geselliger Deutsch - Schweizer Kulturaustausch bei Fondue und schweizer Weissem statt.

Der Kellerclub ist einer von rund 20 Dresdner Studentenclubs. Er wird, wie z.B. auch die Info-Cafeteria allein von Studis geführt. Es ist ein gediegen ausgebauter Keller mit Tanzfläche, Videoraum, reich bestückter Bar und einem Bierausschank. Das ganze in einem grossen Studentenheim. In Dresden wohnen etwa 3/4 der Studis in Wohnheimen, z.T. zu viert in einem Zimmer. Geselligkeit wird hier gross geschrieben und das alles bei einem "Bier für ne Maak". In dieser Beziehung könnte sich Zürich wirklich einiges abschauen.



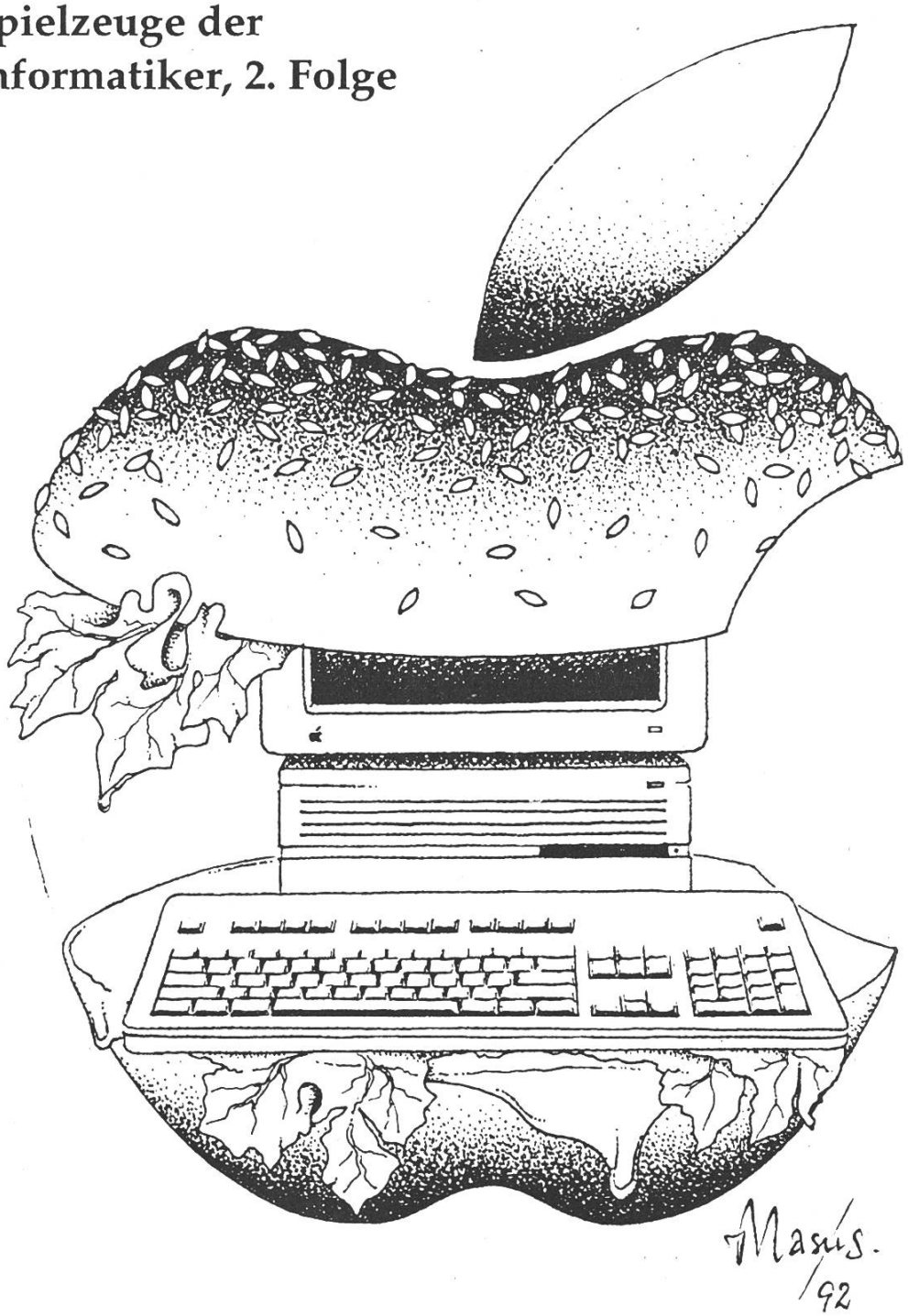
Am Freitag stand dann Sight-Seeing unter der kundigen Führung von Andrea auf dem Programm. Dresden hätte eine Fülle von alten Sehenswürdigkeiten, wäre es nicht am Ende des 2.Weltkrieges in Schutt und Asche gelegt worden. Was nicht wieder aufgebaut wurde (um dann 40 Jahre der Dresdner 2-Takt-Luft ausgesetzt zu werden) liess man wie es war, als "Mahnmal gegen den Krieg" oder ersetzte es durch sozialistische Einheitsarchitektur. Im Moment sind die meissten grossen Bauwerke eingerüstet, um rechtzeitig zur 800 Jahrfeier irgentwann im Jahre 200x fertiggestellt zu werden. Die berühmte Semper-Galerie, erbaut vom gleichen Semper, der auch die ETH entworfen hat, wurde gerade wieder eröffnet. Sie war allerdings für normalsterbliche noch nicht zugänglich, und so mussten wir mit einer Colani-Ausstellung vorlieb nehmen.

Es gäbe noch viel zu erzählen, von einer verrückten Kneipe, deren Inhaber alles sammelt was mit Trams zu tun hat (In seiner an die tausend Exemplare zählenden Tram-Chauffeur-Mützen Sammlung konnten wir auch eine VBZ- und eine BVB-Kappe ausfindig machen. Oder von dem Abend als wir mit Burgund und ihrem Freund durch die Insider-Bars zogen, oder der "dummen Suppe", die uns gegen sibirische Winterstürme gefeit hat. Auch das DDR-Modula-2 Buch das mit den schlichten Worten "Modula-2 gefällt mir" beginnt und mit Ergüssen wie "Pascal ist die Oma von Modula-2" fortsetzt, jedoch den Schöpfer dieser Sprachen mit keinem Wort erwähnt, wäre eine Würdigung wert. Doch es werden nie alle Geschichten erzählt sein und desshalb möchte ich damit schliessen unseren Gastgebern (Tom, Kasi (SCSI?), Andrea, Frank, Burgund+Anhang und allen anderen) ganz herzlich für die 3 Tage Dresden zu danken.

Würden wir ihnen einen ähnlichen Aufenthalt bieten können?

Florian Schlotke

Spielzeuge der Informatiker, 2. Folge



The Big Mac

Ankündigung

Der VIS und das Abteilungssekretariat IIIC freuen sich, Ihnen den Termin der nächsten Kontaktparty "KP 93" bekanntgeben zu können.

Informatik - Kontaktparty

Montag, 25. Januar 1993

14.15 - 17.00 Uhr

in der Mensa Polyterrasse

ETH Zürich

Alle Studierenden der Abteilung IIIC sind herzlich dazu eingeladen.

Der Katalog mit den teilnehmenden Firmen wird anfangs Januar versandt. Für zusätzliche Fragen steht das Abteilungssekretariat jederzeit gerne zu Ihrer Verfügung.

Das Organisationskommitee

Kontaktadressen:

Abteilungssekretariat IIIC
"Kontaktparty 93"
ETH Zentrum
Haldeneggsteig 4
8092 Zürich

Telefon: 01 254 72 11
Telefax: 01 262 39 73

e-mail des VIS: vis@inf.ethz.ch
OK-Präsident: cfranz@iiic.ethz.ch

Telefon: 01 254 7212
Telefax: 01 262 39 73

Perspektiven mit ABB

Praktika für angehende Informatik-Ingenieure und -Ingenieurinnen

ABB Schweiz mit mehr als 35 selbständigen Tochtergesellschaften entwickelt, erzeugt, verkauft und wartet Systeme und Produkte eines breiten Sortimentes zur Bereitstellung und Anwendung elektrischer Energie. Angehenden Informatik-Ingenieuren und -Ingenieurinnen steht damit auch eine breite Palette von Praktikumsmöglichkeiten offen:

System-Software
- Graphische Programmierung
- Compilerbau
- Betriebssysteme

Verteilte Systeme
- Kommunikation
- Prozesssteuerung
- Netzleitsysteme

Datenbanken
- Engineering-Datenbanken
- Nichtstandard-Datenbanken

Wissensbasierte Systeme
- Expertensysteme für Konfiguration und Diagnose

Auf diesen Gebieten arbeiten wir in internationalen Teams an interessanten Projekten. Im Rahmen eines Praktikums haben Sie Gelegenheit, dabei mitzuwirken, persönliche Erfahrungen zu sammeln und Einblick in die Berufswelt zu gewinnen.

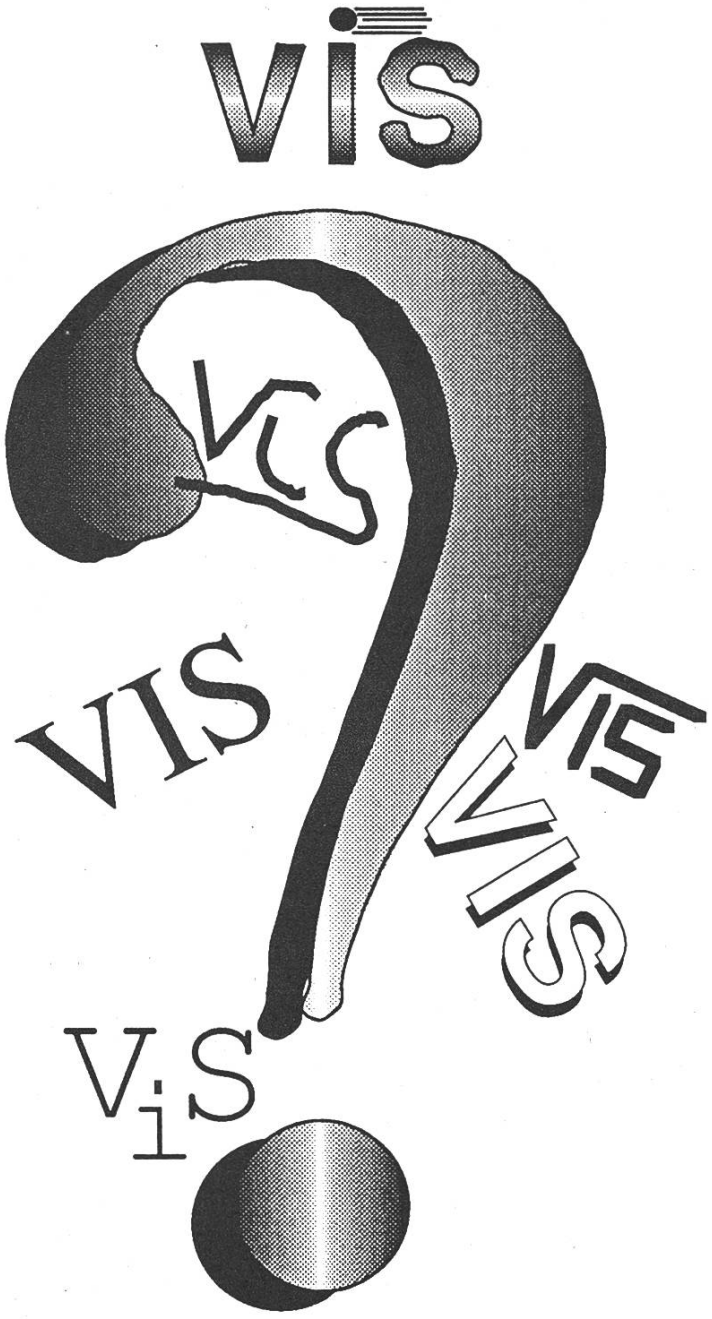
Weitere Auskunft und Unterlagen bei

Ruth Maurer Telefon 056/75 20 56 oder
Dieter Spickenreuther Telefon 056/75 63 31

ABB Management AG
Abteilung PMZ
Postfach
5401 Baden
Fax 056/22 42 26

ABB

Wettbewerb



Worum geht's?

Ihr kennt sicherlich unser altes Logo, den VIS-Würfel. Nun, zum Einen können wir dieses Ding nicht mehr sehen und ausserdem haben wir die

Vorlage verloren[1] und . Es ist also einmal wieder soweit... Der VIS sucht ein neues Logo. Und da wir ein gutes Logo wollen, sollen alle, die sich für kreativ, begabt und extro- intro- oder pervertiert halten, sich etwas einfallen lassen. Wir wählen dann das Beste (das heisst, das was *uns* am besten gefällt, gell?) aus. Und dann? Dann wollen wir es auf T-Shirts, Aufkleber usw aufdrucken! Yeah! Dann haben wir endlich auch so ein T-Shirt mit dem wir uns locker am Strand von Bora-Bora herumflegeln, in Bars Insiderclubs gründen und aus Discos herausfliegen können!

Wie könnt Ihr mitmachen?
Damit allen klar ist, wie die Sache ablaufen soll, sei der folgende Algorithmus dargestellt:

```
Open(Mind);          {Bei einigen Studis leider}
                     {nicht implementiert}
GetFirst(Idee);
while vorhanden(Idee) do
  implementieren(Idee);
  einreichen(Idee);
  GetNext(Idee)      {Es sind mehrere Vorschläge}
                     {erlaubt}
end;
```

Einige Anmerkungen dazu:

- Jede Person kann beliebig viele Vorschläge abgeben, je mehr Vorschläge desto besser. Ideenklau von anderen Logos ist zwar unfein, aber durchaus erlaubt... Wir wissen natürlich von nichts, leben in einer anderen Galaxis etc...

- Die Vorschläge können in beliebiger Form (d.h. elektronisch, auf Papier, posthum, bla bla bla[2]) eingereicht werden. Wir bevorzugen allerdings zweidimensionale, neutral riechende Vorlagen auf DIN A4-Bögen (Lasst also die echt bärenstarke Plastik, die Waldi eben völlig natürlich produzierte wo sie ist). Dreidimensionale *Zeichnungen* sind natürlich OK.

- Die Idee, das Bild einer/s attraktive/n (u.U. noch leichtbekleidete/n) Frau/Mann auf das Logo zu nehmen ist zwar gut und schön, aber weder neu noch originell.

---

[1] OK, *verloren* haben wir sie nicht...
[2] Liest eigentlich irgendjemand diese Texte wirklich sorgfältig (abgesehen vom Redaktor)?

- Die Vorschläge können gerne farbig sein, allerdings solltet Ihr von der Verwendung von spezialfarben wie Gold oder andere metallictönungen Abstand nehmen, da diese sich schlecht reproduzieren lassen. Ihr könnt Bleistift, Scriptol, Acrylfarben, Kohle oder was euch sonst noch einfällt verwenden. Achtet bitte darauf, dass Euer Vorschlag möglichst Kontrastreich wird.

- Eure Vorlagen können beliebig gross sein, doch die beste Grösse ist auf A4 oder A3, denn so können wir sie einfacher verkleinern.

- Die, die bei Open(Mind) bereits mit dem _NotImplemented Trap ausgestiegen sind, sollten sich um einen Assistenzposten bei Prof. [redacted][3] bewerben.

Was springt dabei für mich heraus?
Wir haben, wie gesagt, vor, dieses Logo auf Klebefolien zu drucken. Und auf T-Shirts. Derjenige oder diejenige, die das Logo entworfen hat, bekommt von uns ein solches T-Shirt gratis (auf Wunsch vom Vorstand Handsigniert). Und da ich annehme, dass Ihr Euer Werk mit einer Unterschrift verziehrt, bleibt Euer Name für eine ganze Generation von Informatikern erhalten! Ausserdem wird Dein Bild[4] in einer Ausgabe unserer Visionen erscheinen (womit sollen wir sie denn sonst füllen?).
Ach ja, und vielleicht wirst Du auch zu unserem Mitarbeiteressen eingeladen (wenn wir nicht wegen dem Farbdruck pleite gegangen sind). Und Du bist dir den ewigwährenden Dank Deiner Kommilitonen sicher. Oder so. Brauchst Du wirklich einen Grund, bei so einer Aktion mitzumachen?

Bis wann muss ich kreativ gewesen sein?
Wir wollen zu Beginn des Sommersemesters Nägel mit Köpfen (aber nicht die flache Variante) gemacht haben. Also solltet Ihr die Vorschläge bis zum Ende des Semesters eingereicht haben.

Christian Franz

---

[3] Zensiert... Der Name fängt aber *nicht* mit 'W' an!
[4] Die, die mich kennen, wissen natürlich, dass ich damit nicht ein Photo von Eurer Visage meine, sondern ein Abzug von eurem *Werk*!

Informatik-Projekte zu einem festen Preis?

Normalfall

- Das Software-Haus macht eine oberflächliche Schätzung des Projektaufwands
- Aufgrund dieser Schätzung vereinbaren Auftraggeber und Software-Haus einen Stunden- oder Tagestarif für die eingesetzten Informatiker
- Je länger das Projekt dauert, desto grösser die Einnahmen des Software-Hauses

Festpreisprojekt

- Das Software-Haus schätzt den Projektaufwand aufgrund seiner grossen Erfahrung genau ab und unterbreitet ein Festpreisangebot
- Der Auftraggeber erteilt den Auftrag
- Die Verantwortung für das Einhalten der Kosten- und Terminbudgets liegt beim Software-Haus

Economation AG wickelt seit 20 Jahren erfolgreiche Informatik-Projekte (darunter viele zu einem festen Preis) für namhafte Kunden ab.
An unseren beiden Geschäftsstellen beschäftigen wir insgesamt 35 Mitarbeiter (80% mit Hochschulabschluss).
Wir betreiben eine eigene Entwicklungsinfrastruktur, die Hardware und Betriebssysteme unterschiedlicher Hersteller miteinander verbindet (Schwerpunkt UNIX).

Interessiert? Nehmen Sie mit uns Kontakt auf!

economation

AG für Computertechnik und Automation

Stockerstrasse 46
CH-8039 Zürich
Tel. 01/201 25 52
Fax 01/201 25 56

Freie Strasse 3
CH-4001 Basel
Tel. 061/261 66 01
Fax 061/261 87 32

Einige Hinweise für das Verhalten in mündlichen Prüfungen

Einleitung

Der nachstehende Text ist als eine kleine Hilfe für Kandidaten gedacht, die sich in der Vorbereitung auf mündliche Prüfungen im universitären Umfeld befinden. Zu Beginn wird eine Fallunterscheidung bei der Beantwortung von Prüfungsfragen gegeben, danach folgen einige allgemeine Hinweise und Erfahrungswerte zum Verhalten in mündlichen Prüfungen. Die diskutierten Tips und Hinweise sind explizit als solche zu verstehen, nicht jedoch als Regeln, die neben der vorausgesetzten und unabdingbar nötigen Beherrschung der fachlichen Materie den Erfolg bei mündlichen Prüfungen garantieren. Sie basieren auf Erfahrungen des Autors sowohl als Prüfendem als auch Geprüftem und sind nach bestem Wissen zusammengestellt, jedoch keinesfalls als erschöpfende Liste aufzufassen. Insbesondere kann nicht ausgeschlossen werden, dass Prüfer wie Kandidaten sich völlig anders verhalten, als hier beschrieben, so dass beide Seiten auch weiterhin auf Einfühlungsvermögen und die Fähigkeit, schnell auf den Kommunikationspartner zu reagieren, angewiesen sind. Wenden wir uns zunächst den möglichen Varianten bei der Beantwortung von Prüfungsfragen zu.

Beantwortung von Fragen

1. Die Antwort auf die gestellte Frage ist klar und eindeutig

Möglichst kurz und prägnant antworten, nicht in Details vertiefen, diese werden meist vom Prüfer durch Unterbrechen und Nachfragen abgefragt, wenn nötig. Wenn die Antwort mehrere Punkte umfasst, erst die Hauptpunkte aufzählen, dann erst auf die Einzelheiten eingehen. Dies gibt dem Prüfer die Möglichkeit, jederzeit zu unterbrechen, andererseits hat der Kandidat bereits am Anfang gezeigt, dass er die gesamte Frage beantworten kann, auch wenn der Prüfer die Beantwortung dieser Frage frühzeitig abbricht. Auch wenn der Kandidat der Versuchung unterliegt, das gesammelte Wissen so schnell wie möglich zu reproduzieren, sollte nicht einfach drauflos geantwortet, sondern die Beantwortung so gut wie möglich strukturiert werden.

2. Die Antwort ist eigentlich klar, aber es gibt mehrere dem Kandidaten bekannte Varianten

Dies trifft insbesondere dann zu, wenn mehrere Lehrmeinungen existieren. Es ist von Vorteil, in diesem Fall zuerst die vom Prüfer vertretene Meinung darzulegen (falls bekannt). Ist dies nicht möglich oder nötig, sollte man mit der am häufigsten genannten Lehrmeinung beginnen, jedoch durch einen Satz wie z.B. Hierzu existiert eine Reihe verschiedener Lehrmeinungen ... sofort anzeigen, dass man über die verschiedenen Varianten informiert ist, ohne auf diese sofort einzugehen. Dann kann man zu den

anderen Varianten überleiten, z.B. durch Neben dieser Variante existiert eine Reihe weiterer, insbesondere.... Diese schwache Überleitung gibt dem Prüfer erstens die Gelegenheit, einzugreifen, falls er nur an seiner Variante interessiert ist, andererseits zeigt sie dem Prüfer, dass der Kandidat die Prioritäten so setzt, wie er sie vermutlich erwartet.

3. Es gibt offenbar mehrere Antworten, aber nur eine Antwort ist dem Kandidaten bekannt

In diesem Fall muss der Kandidat seine Unsicherheit bei den Varianten dadurch ausgleichen, dass er so breit wie möglich diejenige Antwort gibt, die ihm bekannt ist, insbesondere entfällt hierbei die unter Punkt 2 genannte Methode, dem Prüfer zu sagen, dass überhaupt mehrere Varianten existieren. Der Prüfer wird die Darstellung des Kandidaten dann höchstwahrscheinlich unterbrechen und durch eine Ergänzungsfrage versuchen, die anderen Varianten genannt zu erhalten. In diesem Fall hilft nur noch die Antwort des Kandidaten, dass er die genannte Variante als die wichtigste/-relevanteste betrachtet und daher ausführlich behandelt hat. Es ist von Vorteil, dann zumindest noch einige Begriffe aus den anderen Varianten nennen zu können, aber wenn das nicht geht, wird der Prüfer es eher schätzen, wenn der Kandidat nicht lange herumprobiert, sondern gleich passt. Dies führt in der Regel noch am ehesten zu einer ausreichenden Bewertung der Antwort.

4. Frage ist nicht klar formuliert (d.h. mehrdeutig)

Hier gibt es drei Möglichkeiten. Erstens kann der Prüfer bewusst versucht haben, den Kandidaten aufs Glatteis zu führen, zweitens kann er tatsächlich aus Versehen die Frage mehrdeutig formuliert haben oder drittens kann der Kandidat die an sich eindeutige Frage falsch (d.h. mehrdeutig) verstanden haben. In allen Fällen sollte der Kandidat sofort nachfragen und um einen Präzisierung der Frage bitten (z.B. durch Könnten Sie diese Frage bitte präzisieren, nach meinem Verständnis könnte sowohl X als auch Y gemeint sein.). In den Fällen 1 und 2 wird dies den Prüfer eher positiv beeinflussen, im Fall drei wird der Prüfer zwar nicht positiv überrascht sein, es ist aber immer noch besser, vorher zu fragen (und dadurch möglicherweise ein Verständnisproblem zu offenbaren), als die Frage falsch zu beantworten und dann vom Prüfer unterbrochen, d.h. noch weiter verunsichert zu werden.

5. Die Frage ist klar, die Antwort ebenfalls, aber der Prüfer fragt dennoch weiter

Hier sucht der Prüfer offenbar nach einem bestimmten Begriff oder einem Sachverhalt, der bisher vom Kandidaten nicht genannt wurde. In diesem Fall sollte der Kandidat dies bemerken, z.B. durch Ich weiss, dass Sie im Moment nach einem bestimmten Begriff in diesem Zusammenhang fragen, bin aber nicht sicher, was Sie meinen. (vermieden werden sollte dagegen die Formulierung: Ich weiss nicht, was Sie hören wollen, o.ä., da hierdurch beim

Prüfer der Eindruck entstehen könnte, der Kandidat versuche, nur das zu sagen, was der Prüfer hören möchte). Dadurch ist der Ball wieder beim Prüfer, der nun seine Frage weiter präzisieren oder zu einem neuen Fragengebiet umschwenken muss.

6. Die Frage ist klar, die Antwort ist aber nicht ganz klar

Hier kann der Kandidat versuchen, zu bluffen, d.h. mit dem vorhandenen Wissensteilen die Frage zu beantworten. Dies ist aber gefährlich, da der Prüfer in der Regel schnell bemerken wird, dass der Kandidat schwimmt und beginnen wird, weitere Fragen zu stellen, die den Kandidaten dann meist entgültig aus der Bahn werfen. Ebensowenig sollte der Kandidat aber einfach passen, d.h. keine Antwort zu geben. Als sinnvoller erscheint es, dem Prüfer zu signalisieren, dass die Frage verstanden wurde, aber der Kandidat sich bei ihrer Beantwortung nicht ganz sicher ist. Dies geht am Besten durch vorsichtige Formulierungen, die die Antwort als nicht vollständig im Sinne des Lehrstoffs, sondern als (unvollständige) eigene Meinung des Kandidaten kennzeichnen, also z.B. Meiner Meinung nach kann diese Frage etwa so angegangen werden: ..., Ich bin hier nicht völlig sicher, aber ich würde die Antwort etwa so skizzieren: ....

7. Die Frage ist klar, aber der Kandidat kann sie nicht beantworten

Dies ist eine für beide Seiten schwierige Situation, die aber eher selten vorkommt (Nr. 6 ist der häufigste Problemfall). Der Kandidat sollte aber unbedingt versuchen, den Schaden so klein wie möglich zu halten und nicht die Stimmung des Prüfers wie auch des Kandidaten auch für die noch folgenden Fragen negativ zu beeinflussen. Also sollte der Kandidat sofort zugeben, dass er es nicht weiss, dies aber nicht so formulieren (eher: Es tut mir leid, aber auf diese Frage bin ich nicht vorbereitet. statt Das weiss ich nicht.). Der Prüfer wird in einem solchen Fall nur die nicht beantwortete Frage als ungenügend vermerken und eine neue, in der Regel nicht verwandte Frage stellen. Falls die Frage nicht völlig neben dem Prüfungsgebiet liegt, sollte der Kandidat vermeiden, mit dem Prüfer über die Fairness/-Rechtmässigkeit der Frage o.ä. zu diskutieren, da dies einen eher negativen Einfluss auf den weiteren Prüfungsverlauf kann.

8. Der Prüfer fragt explizit nach der Meinung des Kandidaten zu einem Problem

Dies ist ein eher schwieriger Fall, für den sich kaum allgemeingültige Regeln aufstellen lassen, da der Prüfer hier nicht nach direkt prüfbarem Wissen fragt, sondern eine Transferleistung, d.h. die Anwendung des vorausgesetzten Wissens auf ein neues Problem, verlangt. Als allgemeiner Hinweis lässt sich aber festhalten, dass die hier geforderten Meinungen besser nicht mit einem Anspruch auf Absolutheit oder vollständige Korrektheit formuliert werden sollten, da dies sofort den Widerspruch des Prüfers herausfordern wird. Weiche Einleitungsformulierungen wie z.B.

Meiner Meinung nach..., Hierzu existieren verschiedene Lösungen, die meiner Meinung nach wichtigste..., Hierzu bin ich der Meinung, ... erzielen hier meist den gewünschten Effekt. Allerdings sollten keine noch weicheren Formulierungen (insbesondere Verwendung von Konjunktiven à la Hier könnte man die Meinung vertreten...) verwendet werden, um den Eindruck von Ausweichen und Unsicherheit zu vermeiden. Die Darlegung sowohl von Argumenten als auch Gegenargumenten ist ebenfalls möglich und wird in der Regel positiv bewertet, der Prüfer wird aber möglicherweise versuchen, den Kandidaten dazu zu bringen, Farbe zu bekennen, d.h. für eine der Varianten Partei zu ergreifen. Hierbei ist eher unkritisch, für welche Variante sich der Kandidat entscheidet, solange er seine Wahl begründen kann, da der Prüfer ja explizit nach der Meinung des Kandidaten gefragt hat. Der Prüfer wird ebenfalls meist nicht negativ (sondern eher neutral oder sogar positiv) bewerten, wenn der Kandidat nicht sofort antwortet, sondern (z.B. durch Dies ist eine sehr komplexe Frage, die ich hier nur im Überblick beantworten kann oder Darüber müsste ich kurz nachdenken.) anzeigt, dass die Frage komplex ist und er sie nicht erwartet hat. Allerdings sollte die Nachdenkzeit einige Sekunden nicht überschreiten. Diese Sekunden sollten aber unbedingt genutzt werden, um die Argumentation zurechtzulegen, da mit Rückfragen und Einsprüchen durch den Prüfer gerechnet werden muss.

9. Völliger Blackout des Kandidaten

Kann vorkommen, insbesondere bei sehr nervösen Kandidaten, ist aber in der Realität eher selten. In diesem Fall sollte der Kandidat diesen Umstand dem Prüfer sofort anzeigen, z.B. durch Verzeihung, aber jetzt bin ich völlig durcheinandergeraten.. Was dann passiert, ist im wesentlichen vom Prüfer abhängig. Kann/will er den Kandidaten beruhigen, so wird er auf ein anderes, meist leichteres Themengebiet umschwenken oder zur gestellten Frage Hilfestellungen geben, bis sich der Kandidat wieder gefangen hat. Hilft dies nicht oder kann/will der Prüfer nicht auf die Situation des Kandidaten eingehen, ist die Prüfung allerdings zu Ende. Dieser Fall ist jedoch fast nicht existent, da die meisten Prüfer es als unfair betrachten, einen Kandidaten auf diese Weise loszuwerden. Oftmals sind die Prüfer auch über Vornoten des Kandidaten informiert und wissen daher meist, was der Kandidat eigentlich leisten könnte, so dass sie versuchen, dem Kandidaten nach Möglichkeit weiterzuhelfen.

MIGROS-GENOSSENSCHAFTS-BUND

INFORMATIK

Die Informatik des Migros-Genossenschafts-Bund plant und entwickelt für die Zukunft!

- Moderne Tele-Kommunikationsnetze für die ganze Migros-Gemeinschaft
- Optimale Logistik- und Lagerbewirtschaftungssysteme
- Effiziente Datenbanken
- Ausgereifte Rechnerverbund-Lösungen
- Experten-Systeme

Sind Sie der ausgewiesene Spezialist,
dann können Sie aus dem Vollen schöpfen und bei der Mitarbeit in einem dieser Projekte einen massgeblichen Beitrag leisten.

Kleine Teams und ein freundschaftliches Arbeitsklima tragen das ihre dazu bei!

Wir freuen uns auf Ihren Anruf, Sie werden alles Wichtige über Ihre zukünftige Laufbahn erfahren.

Unsere Adresse:

Migros-Genossenschafts-Bund
Informatik
Limmatstrasse 152
8005 Zürich
Tel: 01 277 21 11

Weitere Hinweise

10. Unterbrechen des Prüfers

Das Unterbrechen des Prüfers, das Beantworten einer Frage, bevor sie ganz gestellt ist usw. ist zwar ein verständliches Zeichen von Nervosität, kann aber einen negativen Eindruck beim Prüfer hinterlassen. Auch hier gibt es jedoch eine Ausnahme, nämlich dann, wenn der Prüfer so begeistert vom Thema oder seiner eigenen Frage ist, dass er selbst zu referieren beginnt (einige wenige Prüfer versuchen auch, den Kandidaten dazu zu bringen, ihn zu unterbrechen, um die Eigenständigkeit und Selbstsicherheit des Kandidaten zu prüfen). Hier muss der Kandidat zu einem geeigneten Zeitpunkt (z.B. wenn der Prüfer einen Gedanken zuendegeführt hat) den Prüfer unterbrechen, indem er selbst das Wort ergreift. Dies muss jedoch eher vorsichtig geschehen, z.B. durch Wenn ich Sie hier gerade unterbrechen darf... oder, was oftmals besser ist, durch das vorsichtige Äussern einer abweichenden Meinung, z.B. durch Hier kann man allerdings auch eine andere Meinung vertreten... Diese Meinung muss allerdings vertretbar, d.h. nicht ganz abwegig sein und kompetent dargelegt werden, da einige Prüfer Unterbrechungen eher ungnädig aufnehmen, den Kandidaten aber trotzdem eher negativ beurteilen würden, wenn er überhaupt nichts sagt.

11. Unterstützung einer Darlegung durch Skizzen etc.

Oft ist es von Vorteil für Kandidat und Prüfer, Stichworte während der Beantwortung einer Frage auf Papier zu notieren bzw. komplizierte Sachverhalte graphisch/schematisch darzustellen. Der Prüfer wird in der Regel hierdurch nicht negativ beeinflusst, sondern wird diese Form der Präsentation eher zu schätzen wissen. Allerdings sollte er mit dieser Form der Darstellung einverstanden sein. Der Kandidat kann dies z.B. durch die Frage Wenn ich dies einmal skizzieren darf ... testen oder dadurch feststellen, dass der Prüfer auf dem Tisch bereits Papier und Stifte bereitgestellt hat. Diesen Umstand sollte man unbedingt ausnutzen.

12. Formulierungen im Allgemeinen

Zusätzlich zu den bereits genannten Punkten sollten Formulierungen, die Widerspruch erzeugen oder einfach unhöflich sind, unbedingt vermieden werden. Hierzu zählen insbesondere Nein, das stimmt nicht., Aber das ist falsch., Das glaube ich nicht., Das habe ich aber anders gelernt, Prof. XY lehrt das aber anders (Sehr gefährlich!), Da haben Sie unrecht., usw. Dies bedeutet nicht, dass der Kandidat keine gegensätzliche Meinung äussern sollte (manche Prüfer warten fast auf Widerspruch), aber auch hier macht der Ton die Musik. Formulierungen wie Hier lässt sich allerdings auch eine andere Position vertreten oder die Formulierung als Frage Könnte hier nicht auch die Position XY eingenommen werden? sind wesentlich besser geeignet, eigenes Wissen darzustellen, ohne den Prüfer negativ zu beeinflussen.

Es mag nun den Anschein haben, dass Formulierungen allein über das Bestehen von Prüfungen entscheiden können ganz so ist es zwar nicht, aber Prüfer sind auch (nur) Menschen, die eben auch entsprechend menschlich auf das reagieren, was man ihnen präsentiert.

13. Prüfungsdauer

Während für den Kandidaten die Prüfungszeit oft Stunden zu dauern scheint, hat der Prüfer oft den entgegengesetzten Eindruck (ausser, wenn der Prüfer sich über den Kandidaten ärgert, es die letzte Prüfung vor dem Mittagessen oder dem Feierabend ist, oder wenn der Prüfer nicht mehr völlig begeistert bei der Sache ist, weil er die gleichen Fragen und Antworten schon den ganzen Tag gehört hat). Dadurch neigen Prüfer manchmal dazu, länger zu prüfen, als vorgesehen war. Hier sollte der Kandidat vermeiden, durch häufigen Schauen auf die Uhr usw. anzuzeigen, dass die reguläre Prüfungszeit vorbei ist, da er dadurch dem Prüfer Unsicherheit signalisiert. Vielmehr sollte der Kandidat bei den am Ende der Prüfung meist gewechselten Schlussworten (Prüfer: Gut, ich glaube, das wäre dann alles.) kurz darauf hinweisen, dass die Prüfungszeit überschritten wurde (falls eine solche Obergrenze definiert ist). Dies kann z.B. durch eine beiläufige Bemerkung à la Na, es ist ja doch länger gegangen, als ich gedacht hatte. oder Ich hoffe, Ihr Zeitplan gerät nun nicht zu sehr durcheinander. geschehen und kann den Prüfer manchmal bei der Endbeurteilung positiv beeinflussen. War die Prüfung kürzer, als vorgesehen, sollte der Kandidat nicht auf einer Verlängerung bestehen, es sei denn, er fühlt sich ungerecht behandelt. In diesem Fall sollte er darauf hinweisen, dass die Prüfungszeit noch nicht vorbei ist und er glaube, sich durch eine Fortsetzung der Prüfung verbessern zu können. In diesem Fall muss er allerdings damit rechnen, dass der Prüfer nun eher schwierigere Fragen auswählen wird.

14. Sprache und Körpersprache

Auch nichtverbale Signale wie das zustimmende Nicken, Stirnrunzeln (meist Ablehnung, Skepsis), oder das Spielen mit Gegenständen (Taschentücher, Stifte, usw.) können den Prüfungsverlauf beeinflussen. Das Aussenden positiver Signale wie z.B. Nicken, wenn man die Frage verstanden hat, Lächeln, wenn der Prüfer einen entsprechenden Sachverhalt schildert, sollte nicht unterdrückt werden, da es die Prüfungssituation für beide Seiten entspannt und eine Kommunikationsebene zwischen den Beteiligten aufbaut. Auf den negativen Seite beurteilen Prüfer oftmals auch die Haltung eines Kandidaten (aufrecht sitzen, keine Hände in den Hosentaschen!), die Kleidung, Aussprache/Artikulation, Satzbildung (man vermeide Sätze à la: Das ist wenn... oder unvollständige Sätze, bei denen der Prüfer Mühe hat, den Sinnzusammenhang zu erkennen) usw. Zwar weiss jeder Prüfer, dass Kandidaten in der Regel nervös sind, aber

diese Nervosität sollte nach Möglichkeit nicht den Prüfer anstecken. Daher sollte der Kandidat vermeiden, mit Gegenständen wie Schlüsselbunden usw. herumzuspielen, mit Kugelschreibern zu klicken, Papiertaschentücher zu zerfetzen etc. Wenn es sich absolut nicht vermeiden lässt, sollte der Kandidat mit etwas spielen, was sich der Sicht des Prüfers entzieht und keinen Lärm macht (z.B. Drehen an Fingerringen). Eine schwierige Frage ist diejenige des direkten Augenkontaktes. Während sich manche Prüfer und Kandidaten damit schwertun, kann er jedoch auch von Vorteil sein und eine gewisse Persönlichkeit/-Selbstsicherheit des Kandidaten signalisieren. Zumindest bei der Begrüssung und bei der Verabschiedung sollte der Kandidat direkten Augenkontakt aufnehmen (also bitte keine Sonnenbrillen etc.), es sei denn, der Prüfer sei bekannt für seine Antipathie gegen direkten Augenkontakt.

Schlussbemerkungen

Einem Missverständnis sollte hier noch vorgebeugt werden: die vorliegende Liste von Hinweisen ist keineswegs als Versuch gedacht, möglichst höfliche, dem Prüfer genehme, Wunsch-Kandidaten hervorzubringen. Kritisch denkende, selbstständige Studenten sind das Rückgrad einer Hochschule und der aufrechte Gang in Prüfungen macht sicherlich mehr (positiven) Eindruck als eine deutlich sichtbare Bereitschaft zur unbedingen Anpassung. Wie in den meisten anderen Lebenslagen auch, kommt aber dem eigenen Auftreten in Prüfungen einige Bedeutung zu, zumal mündliche Prüfungen explizit nicht nur das gelernte Wissen des Kandidaten abprüfen sollen, sondern auch seine Fähigkeiten zur Anwendung seines Wissens auf verwandte Gebiete, zur kritischen Meinungsäusserung usw. auf die Probe stellen. Der persönliche Eindruck des Prüfers von einem Kandidaten wird in jedem Fall einen gewissen Einfluss auf die Prüfungsnote haben.

Abschliessend sei nochmals darauf hingewiesen, dass nicht so überzeugend auf einen Prüfer wirkt, wie die Beherrschung des Prüfungsstoffes und dass die Befolgung der vorgestellten Hinweise keinesfalls eine Garantie für das Bestehen von Prüfungen ist. Es kann jedoch festgehalten werden, dass der Prüfer in der Regel kein sonderliches Interesse daran haben wird, einen Kandidaten als ungenügend zu bewerten, da er ihn in diesem Fall meist nochmals prüfen muss und da eine hohe Quote von durchgefallenen Kandidaten auch für den Prüfenden (= Lehrendem) kein gutes Zeugnis darstellt.

Hannes P. Lubich, ETH Zürich, 1992

Praktikumsberichte UBILAB

Ich absolvierte den zweiten Teil des Praktikums à 7 Wochen in den Sommerferien vom 20. Juli bis 4. September bei der UBILAB. Mein Job war allerdings ein etwas aussergewöhnlicher. Das Team, bei dem ich angestellt was, hatte ein besonderes Projekt mit dem Namen 'kap92' in Angriff genommen. 'kap' bedeutet 'Konzernbetreuungs-Arbeitsplatz'.
Dies ist eine auf einer Sun-Workstation angebrachte Arbeitsumgebung, die es erlauben sollte, verschiedene Büroanwendungen nebeneinander zu benützen und miteinander zu verbinden: Textverarbeitung, Tabellenkalkulation, Text-Retrieval in Fachzeitungen etc. und vor allem eine eigens entwickelte Datenbankapplikation. Diese Applikation stellt den eigentlichen Kern des Projekts dar, sie vor allem sollte die Arbeit der Konzernbetreuung (SBG-Abkürzung KOKO) erleichtern und bildet den Stolz der Entwickler. Mein Job hatte aber nichts mit Entwicklung zu tun. Meine Aufgabe war es, Benützer einzuführen und je nach Notwendigkeit zu schulen. Was mich an der Aufgabe reizte, war die Idee, dass ich die Probleme, von denen in der Vorlesung 'Arbeitspsychologie' erzählt wird, einmal in Realität antreffen würde - und auch der Gedanke, dass ich sehr vieles kennenlernen konnte. Zum ersten Punkt gibt es noch zu bemerken, dass dies auch die UBILAB-Leute interessierte, die (zum Teil) an der ETH studiert hatten, einer davon im Nebenfach Arbeitswissenschaften. Deshalb war AOC (nein, dies ist kein gesponserter Bericht) angestellt worden, eine begleitende Untersuchung zu führen. Während meines Praktikums wurde die Sache einem Teil der Angestellten als Pilotprojekt vorgestellt, wobei die Teilnehmer am Pilot durch ihre Chefs bestimmt worden waren ('wir machen mit').
Am Anfang musste ich die Angelegenheit natürlich selbt lernen: das waren zuerst das 'X.desktop', einige Dinge wie die Bedienung eines bewunderswert benutzungsunfreundlichen Mailtools und dann v. a. Text- (und Graphik-) Verarbeitung mit FrameMaker. Dazu wurde für 3 weitere Angestellte und mich eine externe Lehrerin angestellt, die uns vier Tage lang unterrichtete und damit auch schon fortgeschrittene Frame-User beeindruckte.
Dann begannen die Schulungen der KOKO-Mitarbeiter, und zwar bei den SachbearbeiterInnen. Mein erstes 'Opfer' kannte schon Windows, mein zweites MacIntosh. Dies bedeutete für mich, dass ich die graphische Oberfläche nicht speziell erklären musste, und dass Aktionen wie 'drag and drop' schon bekannt waren. Da wir 'Frame-Gurus' aber noch nicht sehr viele Templates vorbereitet hatten, konnten wir den ersten Benutzern in dieser Beziehung noch nicht allzu viel

präsentieren. Bei den nächsten Schulungen waren wir dann hier etwas weiter, doch dafür waren andere Probleme grösser: beide Sekretärinnen hatten noch nie eine graphische Oberfläche gesehen ('linke Taste drücken, drauf bleiben, schieben, loslassen...') und waren der ganzen Sache gegenüber eher skeptisch eingestellt (davon berichtet auch die Voruntersuchung der AOC). Die Hoffnung des Schulungschefs war u.a., dass die Vorstellung 'von Frau zu Frau' einige Berührungsängste abbauen würde - über den Erfolg dieser Strategie kann ich nicht urteilen. Den Sekretärinnen musste ich v. a. die Textverarbeitung mit FrameMaker nahebringen, was nicht so ganz einfach war und ist. Das Arbeiten mit Formaten ist doch eine ganz anderes Vorgehen als 'nur' korrektes Abtippen von Sudelvorlagen per Schreibmaschine. Meine Beobachtungen ergaben auch, dass sich die Sekretärinnen einen komplizierten Arbeitsablauf merken (oder jedenfalls aufschreiben und nachher ausführen) können, sich aber nur zurückhaltend mit Strukturen wie hierarchische Directories, Absatzformate, Schablonen etc anfreunden.

Der nächste grosse Teil der Schulungen betrifft die Einführung der Datenbankapplikation, die allerdings noch nicht ganz betriebsbereit ist. Die Applikation wurde mit 'Ingres - Windows-4GL' erstellt (dabei haben sich auch schon ein paar Kollegen im Code verewigt). Ich hatte wenigstens die Gelegenheit, eine dieser Viertgenerationssprachen, von denen ich bisher auch nur in der Vorlesung 'Informationssysteme' gehört hatte, aus der Nähe zu bestaunen.

Windows-4GL erlaubt es in relativ kurzer Zeit mehr oder weniger hübsche 'Frames', i. e. Masken, die auf Eingaben mit Erzeugung von allerhand Events reagieren, zu erstellen und deren Felder und Tabellen über Variablen mit Datenbankabfragen zu verbinden. Die Sprache wird (von Ingres) objektorientiert genannt, die Objekte sind aber nicht ganz (Informatik 4-) echt. System-Klassen haben Attribute, Methoden und erkennen Events (Mouse-Clicks etc), auf welche Reaktionen angegeben werden können; Benutzerklassen haben nur Attribute. Erweiterbare Objekte gibt es nicht. Die Kommunikation zwischen verschiedenen Frames wird schnell ziemlich komplex und ist bei einem umfangreicheren Projekt nicht ganz einfach handzuhaben (und ist hier bis jetzt auch noch nicht ganz wunschgemäss realisiert).

Bis jetzt wird die Applikation als 'β–Release' einigen SachbearbeiterInnen vorgestellt, die die Korrektheit der Daten überprüfen und v. a. allfällige Verbesserungsvorschläge und weitere Wünsche anbringen sollen. Ich selbst kann mir die Sache von 'innen' (Code) und von 'aussen' ansehen - damit ich auch verstehe, wo welche Veränderungen angebracht werden können. Von den bankfachlichen Hintergründen verstehe ich immer noch ziemlich wenig - bis zu meinen

nächsten Schulungen muss ich da noch einiges lernen.
Alles in allem habe ich ein für mich sehr interessantes Praktikum gemacht, dazu beigetragen haben auch meine Kollegen hier - sowie von der UBILAB als auch von der KOKO.

Susanne Werner IIIC/7

Während den Frühjahrs- und Sommer-Ferien absolvierte ich mein Industriepraktikum im UBILAB der SBG. Da ich schon vorher im UBILAB als Werkstudent arbeitete, brauchte ich mich nicht ans neue Klima zu gewöhnen.

Die Aufgabe des ersten Praktikumteils bestand darin, zu einem von einer externen Firma hergestellten Document Server eine Benutzerschnittstelle zu schaffen, die eine transparente Navigation in einem in einer relationalen Datenbank gespeicherten Dokumentraum ermöglicht. Die Hilfsmittel waren – verglichen mit dem Umfang des Problems – bescheiden: eine Sparc-Station mit lokalen root-Privilegien :-), eine C-Bibliothek zum Doc Server und eine RPC-basierende Bibliothek zu einem grafischen Desktopprodukt, das so schlecht war, dass ich seinen Namen hier nicht erwähnen möchte... Da die Aufgabe sehr weit gesteckt war, musste ich erst mal entscheiden, wie das Problem am besten anzupacken sei. Ein vielversprechender Ansatz war die Einbettung des Document Clients ins UNIX Filesystem. In der ersten Phase experimentierte ich mit Filessystemdrivern, die Kontakt zur Datenbank hatten und einen Teil meiner Testdokumente in Directories abbilden konnten. Dieser Ansatz hat einen konzeptionellen und einige technische Nachteile: Die logische Abbildung des Dokumentraums war für die hierarchische FS Struktur nicht vorgesehen – mir fehlte schlichtweg eine Dimension. Auf der Technischen Seite war es nicht allzu ratsam hochkomplexen und auch nicht speziell stabilen Code in den UNIX Kernel zu linken, die Folgen sind wohl jedem klar (Dazu muss angemerkt werden, dass sowohl DocServer wie auch Desktopprodukt ihre Schnittstellen noch in der Entwicklung hatten). Viel später, nachdem ich den Ansatz Filesystem schon verworfen hatte, kam mir noch die rettende Idee, wie die technischen Hindernisse erfolgreich ausgeräumt werden könnten: Mit einer Verlagerung des Filessystemdrivers von der Client auf die Server Seite hätte man bestehende, stabile FS-Clients brauchen können und den Datenbanklink in Form von NFS Servern realisieren können, die nicht im Kernel laufen sondern als ganz normale *daemons* auf einer andern Maschine sitzen. Um nicht weiter mit Strukturproblemen kämpfen zu müssen implemtierte ich schliesslich das Problem in Form eines Desktopservers, der die spezielle Struktur des Dokumentraums visualisieren konnte. Abgesehen davon, dass ich mit relativ

vielen Fremdprodukten kämpfen musste, war die Arbeit sehr lehrreich. Dass die Problematik des Document Managements sehr komplex ist und dass auch forschungsmässig noch einige Arbeit geleistet werden muss, zeigte sich auch in diesem konkreten Projekt: Im Moment wird es von einem Kollegen im Rahmen einer Semesterarbeit weiterverfolgt.

Der zweite Teil des Praktikums absolvierte ich im selben Projektteam, wobei ich eine ganz andere Aufgabe erhielt. Ich sollte zeigen, ob sich zur Entwicklung von komplexen Datenbankapplikationen OOP-Werkzeuge eignen und wie sie sich im konkreten Fall mit 4GL Tools vergleichen lassen. Beeinflusst von der OOP Vorlesung war für mich die Anwort schon vor Abgabe der Arbeit klar, weshalb auch die Motivation entsprechend hoch war. Die 4GL Tools, mit denen Kreditapplikationen entwickelt wurden stammten von Ingres, liefen unter X11 und benutzten natürlich auch die Datenbank von Ingres. Die OOP Tools waren das NeXTSTEP AppKit (eine grosse Sammlung von Objekten, die die schnelle Entwicklung von Applikationen mit grafischer Benützeroberfläche erlauben) und das damals neu erschienene DBKit, das eine datenbankunabhängige OO-Schnittstelle anbietet. An meinem Arbeitsplatz hatte ich eine NeXTStation Turbo Color zur Verfügung und konnte auf eine Sun 690MP als NFS- und DB-Server zugreifen. In der ersten Phase arbeitete ich mich ins DBKit ein – wobei ich erfreut feststellte, dass im Gegensatz zu ESQL mit sehr mächtigen Objekten jongliert werden konnte. Gleichzeitig nimmt aber auch die Fehlerquote rapide ab, da die Problemlösung meist sehr einfach und präzise formuliert werden kann. Die Einarbeitung in die Sprache Objective-C die von NeXT verwendet wird, war sehr leicht. Objective-C ist eine kleine, Smalltalk-ähnliche Erweiterung von ANSI C und überhaupt nicht mit dem komplizierteren C++ vergleichbar. Innerhalb dieser Startphase produzierte ich einige Objekte die später in der Kreditapplikation verwendet werden sollten. So brauchte es spezielle Tabellenansichten die sich automatisch sortieren konnten, es waren Objekte zur Gruppierung von Interface Elementen nötig, es sollte spezielle Drag&Drop Mechanismen zum Übertragen von Records geben etc. Diese Art von Objekten wiederum wurden zu Paletten verschmolzen die zum einen Funktionalität konzentrieren und zum andern eine saubere und codelose Einbindung in die Applikationen erlauben (mit Drag&Drop und Interface Browser). In der zweiten Phase war eine Klassenhierarchie zu kreieren, die sich jedoch im Laufe der Zeit durchaus ändern konnte. Solche Änderungen waren z.B. das Zusammenziehen von ähnlichen Klassen unter abstrakten Superklassen. Mit der Zeit lernte ich vorallem das DBKit schätzen, weil es einen kompletten Satz ausgefeilter Objekte anbietet, die einen bei der Arbeit mit rela-

tionalen Datenbanken selten in Verlegenheit bringen. Genauso wichtig ist, dass man Erweiterungen sehr einfach selber Einbetten kann, wenn man ab und zu mit seinen Wünschen über die Möglichkeiten des Kits hinauswächst. Subclassing ist die eine Möglichkeit, das Bilden von horizontalen Categories die andere, die dadurch besticht, dass sie alle vorhandenen Standardklassen ohne Sourceänderung mit der neuen Funktionalität erweitert. Um die entstehende Datenbank-Applikation auch Performancemässig vergleichen zu können, sollten sowohl die OOP Version wie auch die 4GL Variante auf dem selben Ingres DB Server operieren. Da das DBKit relativ neu war, mussten wir auf den Ingres-Link etwas länger warten, was zu einem interessanten Portabilitätstest führte: Die gesamte Entwicklung wurde vorerst mit einem äquivalenten DB Modell auf der SYBASE Datenbank begonnen um später durch Änderung *eines* Strings auf Ingres zu wechseln (BTW: dieser String steht nicht mal im Source Code, er ist in einer externen Resource à la Mac eingebettet und kann auch zur Laufzeit geändert werden, was zu ganz besonderen Effekten führen kann...). Da ich zu Beginn schon mit SYBASE und ORACLE solche Tests machte, war ich relativ gelassen, als das Ingres Adaptor Binary erst 3 Tage vor der Abschlussdemo als NeXTmail bei mir eintraf. Zum Schluss konnte ich dann in der Praxis zeigen, wo die stärken des OOP Ansatzes lagen: Sehr wenig Code ist nötig, um dieselbe Funktionalität zu erreichen wie mit einem 4GL Werkzeug (15-25%), durch die kompakte Formulierung komplexer Probleme reduziert sich die Entwicklungszeit stark (Faktor 2..3, wobei hier fairerweise die Analyse ausgeklammert werden muss) und durch einfache Erweiterbarkeit kann man die Tools nach seinen Wünschen ausrichten (während ich den Eindruck einfach nicht los wurde, dass die 4GL Entwickler sich nach dem Tool richten müssen). Neben diesem Schwerpunkt OOP hat mir das Praktikum viel auf dem Gebiet der relationalen Datenbanken gebracht und ich hatte einen sehr umfassenden Einblick in eine aktuelle Client-Server Umgebung.
Bleibt nur noch, mich beim KAP92 Team und dem ganzen UBILAB Team für die intensive Betreuung zu bedanken, denn ein Praktikum kann ja auch Spass machen, oder nicht?!

George Fankhauser IIIC/6

Should Software Be Copy Protected Part II

The Debate

Essentially there are two sides to this issue: those persons who are in favor of software copy protection, and those against it.

Side One

Arguments

Software developers and publishers are the principal supporters of software copy protection. They believe it is necessary to protect their businesses because software piracy costs their industry millions each year through lost sales. Since most illegal software copying is committed by everyday software users, developers feel that copy protection is worthwhile because most users will not be able to defeat the protection. Some developers feel that as long as there is a way to steal software, some people will do it.

Analysis of arguments

From the previous description of the various copy protection strategies, it is apparent that they, by design, restrict or limit the way software is used. There are two goals behind this: (1) to insure that only the legitimate purchaser of the software can use it and; (2) to deter the purchaser from giving a copy of the software away.

Of course, the ultimate goal of the software developer is the sale of more software. It is not known however, if software copy protection actually increases sales; some argue that it does not, and in fact, may decrease sales. This effect is caused by users purchasing other competing software that may function similarly but is not protected.

Further, the effect of software piracy on profits may be exaggerated. Often the effect is estimated by multiplying the number of illegal copies in circulation by the sales price of the software. However, this assumes that those persons who have pirated software would actually have bought it. For instance, suppose a software developer sells a software program for $75 and has sold 75 copies to users. Also suppose that 25 persons, for whatever reasons, have illegally obtained copies of the program. It would seem that 25 people have stolen the program and that the developer is losing $75 times 25 copies or $1,875, which represents a 33% loss in revenue. The developer then decides to invest in the development of some form of copy protection. But instead of those 25 people buying the program, only 5 people using the pirated copies value the program enough to buy it. The developer gains just $75 times five or $375 in revenue, which is a 7% gain. More importantly, very little of this gain translates into profits because of the costs incurred to develop the protection. (Zagorsky 21).

The Office of Technology and Assessment (OTA) reported in 1990 that the loss estimates reported may be "subject to bias" because the reports are published by companies or agencies with ties to the software industry (Zagorsky 21).

In discussing the upside of software piracy, Jay Zagorsky argues that software piracy actually has a positive effect on future software sales because persons who pirate software often buy a legitimate copy once they have determined its value to them. Furthermore, additional sales are

also generated when pirates buy tie-in products that enhance the abilities of their illegal copy. Another factor that should not be overlooked is that even though piracy does not add to current sales, it does provide exposure that may boost future demand for a program (Zagorsky 21).

Values

The primary values held by those who think computer software should be copy protected are economic prosperity and economic security. These are basic values held by everyone in business. Software developers are no different in that respect; they are in business to make money. As evidenced by the published reports, piracy has a definite impact on the software industry. Computer software is costly to develop and market, and software developers want that investment protected.

A secondary value held by software developers is pride in their work. It takes a great deal of time, energy and ingenuity to develop a piece of software. Software developers feel that theft of their product is a personal violation, just as the burglary of a home is a personal violation. Software developers consider it is a compliment when someone purchases their software product (Rosenberg 47).

If the primary value held by this group is economic security, then the cost of developing the copy protection scheme is not the only thing a software publisher has to consider when deciding whether to copy protect their product. There are other factors involved, such as the increased cost of providing maintenance to legitimate users who encounter technical problems with the copy protection. Most of the technical problems are a matter of convenience rather than disaster, but the costs are real and have led some software developers to abandon copy protection altogether (Rosenberg 46).

Side Two

Arguments

Opponents of software copy protection are the users of computer software. This group includes private individuals, businesses, academic institutions, and even the software developers themselves. The arguments behind their positions are many, but, unlike software developers, not much has been published regarding their views on this subject. For this reason, many of the arguments discussed here were obtained from an informal survey conducted by this researcher over the Internet, a worldwide computer network concerned primarily with the exchange of information. The solicitation that was used is shown in Appendix A.

The primary reason that users argue against copy protection is that they feel that most copy protection schemes are annoying and interfere with the normal use of software that has been legitimately purchased. To reinforce this argument, most users cite the various problems they have encountered from the various copy protection schemes.

With "master disk" type of protection, users cannot make a backup or archival copy of their software. If the software's master disk becomes damaged or unreadable, they must request a replacement. This usually involves mailing the defective master disk to the publisher along with a receipt. While they are waiting for a replacement, they cannot use the software. Master disk protection also limits the software's use because users are not able to install it on a hard disk drive. Instead, the software must be run directly from the master disk. For

these reasons master disks are the most hated type of protection.

Many users find authorized "key disks" a nuisance. Recall that this type of protection requires that the key disk be inserted in the disk drive every time the software is run. Also, just as master disks, if the key disk becomes damaged or unreadable, the software will not work. The user must send a request along with proof of purchase to the publisher for a replacement.

Users dislike dongles because they are another piece of hardware that must be attached to their computer. Dongles may also occupy the place of some hardware component the user needs. Many personal computers are limited in their expandability and cannot support the extra hardware device. If many developers used dongles to protect their software, users would have to have several dongles attached to their computer, one for each software package.

The "code lookup" form of protection has received mixed responses from software users. Some find it an acceptable form of protection because it allows the software to be backed up and installed on a hard disk drive. Others feel that having to look up and enter the access codes is a nuisance. Also, long term reliability is at question because if the code sheet gets lost or destroyed the software is no longer usable. Some users transcribe the access codes on a plain sheet of paper as a backup measure.

The recent OIDS-style protection has also received mixed responses from users. Recall that a program using this type of protection is "locked" on a master disk until a user registers it. Most users find this acceptable because, once the program is registered, it is totally unprotected, allowing them to use the program however they please. Others feel it is a hassle to have to send in the registration card in order to get the necessary authorization code to remove the protection.

Users do not voice any arguments against the authorization code and registration screen types of protection, because they are primarily deterrents and do not interfere with the use of the software. Some users like registration screens because their copy of the software can be personalized with these screens.

In addition to the problems found with the protection schemes themselves, most users argue against the very existence of copy protection. Since real pirates can get around any copy protection scheme, the reason for having copy protection at all is nullified. Copy protection effectively serves no purpose and only hurts those people who are honest. Also, it only serves to make software more expensive and the development time used to invent the protection scheme is wasted.

This view is exemplified by the survey response given by Robert Dorsett. He states, "It [copy protection] needlessly penalizes legitimate users. . . . It adds unnecessary expense to software, provides unneeded support to the industry leaches [sic] which market copy protection strategems, and, in the long run, won't affect sales much. People who buy software will buy software; those who won't, won't" (Dorsett).

Tom Johnson concurred that, "As a developer and a user I am completely and totally against copy protection. I have had to implement it a couple of times . . . Development time increases, support increases, and it doesn't really seem to help that much in the long run--the pirates just patch out the copy protection. . ." (Johnson).

Bob Gross, a software developer for Molecular Biologists, disagrees with this view stating that the copy protection

scheme his company uses does not add any cost to the software (about $.50 per disk), but it protects their interests and allows them to continue development (Gross).
Finally, users feel copy protection leads to unreliable computer systems, since it usually employs some type of alteration to the system software of a computer. Several survey respondents gave detailed accounts of how copy protected software they had purchased quit working as the result of upgrading their computer to the latest system software.

Analysis of arguments

Although users argue against the existence of copy protection, they fail to recognize the ties they have to the software developer. Since software piracy can weaken or bankrupt a software developer, the users can in turn be harmed, as they may have to pay more for software or do without it altogether (Wiggins 33).
Some software users argue that they are not really pirating software but "testing" it for some period of time. This reduces the risk in purchasing a program, because it allows them to evaluate competing products and purchase the one that best fits their needs. This argument seems reasonable but is not when compared to the purchase of other products. There are few products, if any, that can be tested over an extended period of time without being purchased first. For example, people cannot walk into their local Ford and GM car dealerships and ask that each dealer give them a car to drive for free, until they decide which car they wish to purchase.
Users can learn whether or not a program works as advertised or meets their needs by doing a little research before purchasing. It is possible to learn a great deal about a software program by reading reviews in magazines, by attending user group meetings, by getting opinions from friends familiar with the program, or by getting a demonstration from a local dealer. Further, many mailorder houses allow software to be returned within 30 to 90 days if it does not meet the needs or fulfill the expectations of the purchaser.
Extended trial periods have been tried by some of the smaller software developers. Software that is released under the "shareware" concept gives users the opportunity to use a program for some period of time and, once they have determined that it meets their needs, pay the requested shareware fee (usually $5 - $50). However, the "shareware" idea has been largely unsuccessful. Most shareware developers report that few people actually pay their registration fees.

Values

The opponents of copy protection base their arguments on the values of efficiency (convenience), security, fairness and personal integrity, with emphasis on efficiency. As stated by their arguments, users feel that there should be no restrictions on how they use a program that was legitimately purchased. Most copy protection schemes (by design) are limiting agents. They restrict how a program is used and these restrictions are annoying to most users.
In addition, some copy protection schemes do not allow backup copies to be made of the software. Some users place a high value on security. What will they do if the copy protected disk which contains software vital to their business gets destroyed or somehow becomes unreadable? How long will it take to get a replacement disk?
The value of fairness is considered with regard to the money spent to purchase a

program. Users feel that a computer program should work properly and be worth the money it took to purchase it. Most users feel that software is too expensive and that this is the real cause of the software piracy problem. Conner and Rumelt concur with this view. In their paper Software Piracy: An Analysis of Protection Strategies, they state that "one simple strategy always open to firms that wish to reduce piracy is to lower price" (Conner 129). They maintain that if the price of a software program is less than or equal to it's perceived value or the cost involved with pirating it (possible penalties or lost reputation), people will buy the program instead (Conner 128).

This view is not universally accepted however. Robert Wiggins states that some of the most often pirated software costs less than $30. This includes "nonessential" software products such as video screen savers and games (Wiggins 33).

Finally, copy protection is considered an insult to those who purchase software because it presumes guilt. Many users prefer to buy software from companies whose protection policy is based on mutual trust, "We trust you, and are not copy protecting this software. Please don't give copies away" (Long).

Rodney Jacks
Professors Winters and Zacchaeus
RCM 60.02
12/4/91

Newsgroups: comp.security.misc
Subject: A Parable of our times, The three little Pigs
Sender: news@dxcern.cern.ch
Organization: CERN European Lab for Particle Physics
Date: Mon, 23 Nov 1992 14:33:47 GMT

A Parable.

Once upon there were there were three little pigs who following the dictates of the narrative built three houses of straw, wood and brick. All were fully aware of the dangers of canine predation and took security measures accordingly.

The brick house Pig was a government contractor who decided that the only safe way to live was to build a thick brick wall with no doors or windows and a steel roof. Testing this against a three megatonne huffer and puffer he was confident against attack by all pig eating wolves in the neighborhood.

Having less money the straw and wood house Pigs had to work for a living. They thus had to have doors and windows but used a clever method of tightly interwoven strands which prevented attack if done properly. The straw house pig was so proud of his that he had the outside painted in a clear varnish so that everyone could see how beuatiful it was. He also had an old wood burning stove from an old farm cottage installed together with an old oak dresser in which he stored his boxes of high fibre Muselli.

The Wood house Pig was rather more practical and had the tightly woven strands painted in thick black tar. "You never know, it may do some good" he said, "and the nice thing about tar is that if the wolf comes and tries to tamper with it I might just spot a paw print or something".

Straw house pig was very upset by this and taunted wood house pig about relying on "security through obscurity". Wood house pig could not understand the fuss that straw house pig made. After all if anybody did ever figure out a way of undoing tightly woven strands at least he had his tar to fall back on, and if someone started unpicking the tar they would have to be quite carefull if he wasn't going to spot them doing it.

When the big bad wolf came along it turned out that he was not as expected the grey Wolf commonly seen in those parts but a brown one. Having studdied the behaviour of Porcus Mensae (Housebuilding Pigs) he was familliar with the principle of tightly interwoven strands and knew it to be a formidible defense. He also knew that the Pigs had subcontracted the housebuilding to some of the local villagers who were not always carefull about the way in which they wove the strands. Because the straw house pig was so proud of his achievements and

displayed the intricate weftwork for all to see the Wolf quickly spotted a crucial flaw in the construction and began his huffing and puffing routine.

After the house had been huffed and puffed down and the straw house pig had run off to the wooden house the Wolf surveyed the Wooden house. Because the weft work was coated with sticky tar it was very hard to see the pattern of the weave and spot the weak point.

Undaunted the Wolf appeared at the door dressed as a pig "Hello I'm brick house pig, the Wolf has blasted the side of my house away with a thermonuclear device, can I come in?" Wood and Stwaw house pig were so pleased that their efforts had been more successfull than brick house pig that they opened the door to be greeted by the Wolf.

Banging on the door of Brick house pig, Straw and Wood house pig shouted, "Please let us in, he's going to eat us". Brick house pig was however on holiday in the South of France at the time and so didn't see Straw and Wood house pig being roasted alive on his doorstep.

Dismissing simple security precautions because of an ideological belief is plan stupid. Computers should be as secure as possible when supplied. If the machine breaks down, it is the vendors fault. If the machine is broken into using an attack which has been known for a long time an that the user has not been warned about it is the manufacturers fault.

UNIX vendors are simply *NOT* going to sell their systems unless they shape up. As supplied UNIX is insecure. It is not the customers responsibility to patch an operating system for such obvious flaws.

The Password File should always be protected from unauthorized access.

System accounts should always have two passwords.

Passwords should be checked against a dictionary.

Default accounts with system privileges should be disabled.

G.A.B. 6648 Minusio

Falls unzustellbar bitte zurück an:

Verein der Informatikstudierenden
IFW B29
ETH–Zentrum

CH–8092 Zürich

Inhalt