APL-Journal

A Programming Language

1/2001

In dieser Ausgabe:

David Liebtag
Using Namespaces: A Practical Example

Axel Güth
APL2 für Windows:
Verbesserungen im Service Level 8

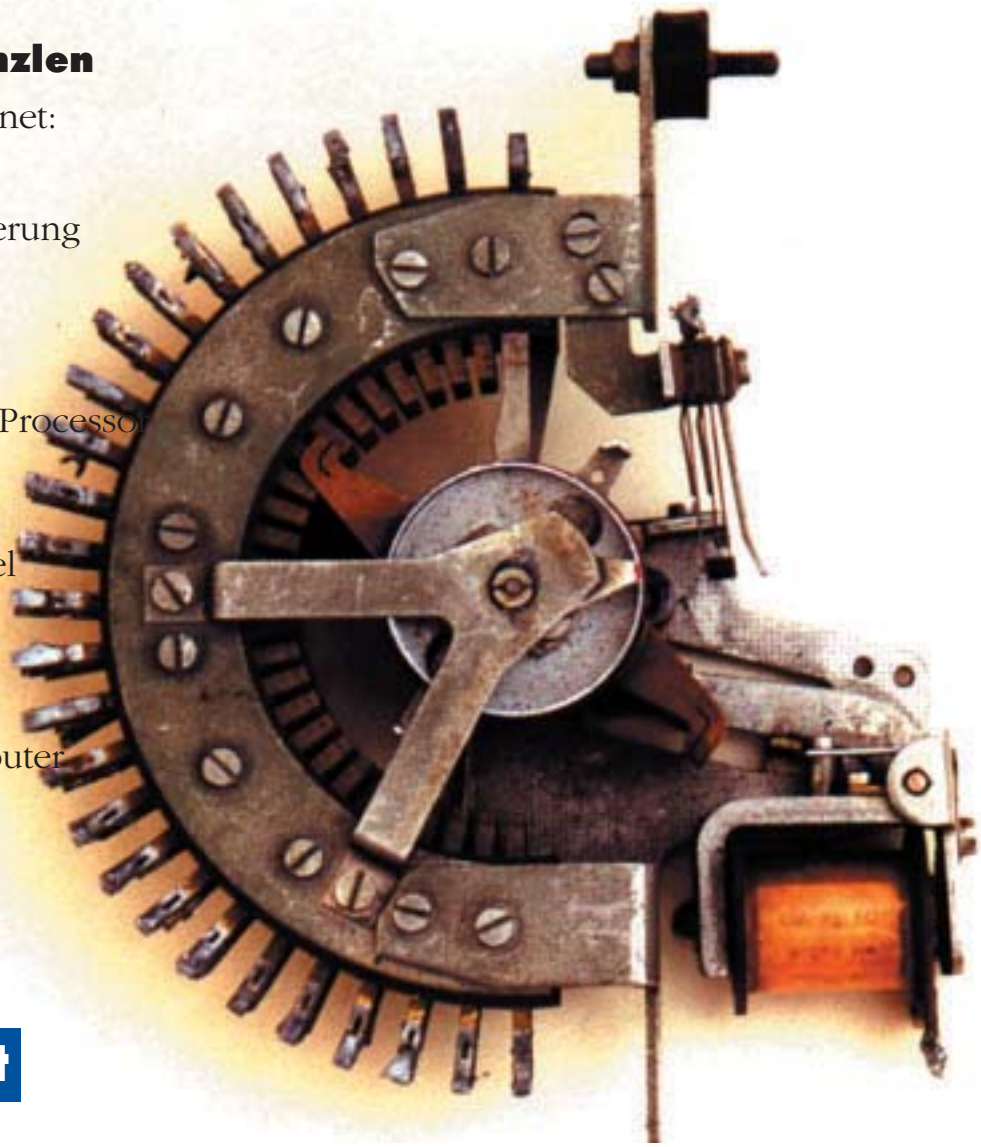
Conrad Hoesle-Kienzlen
Vom Einzelplatz ins Internet:
Die Fallstricke bei der
Client/Server-Programmierung

Herbert Voß
LYX
Open Source Document Processor

Dieter Kilsch
Durchdringung von Kugel
und Zylinder

Wolf Dehus
Opa und der neue Computer

Nina Kroll
Frauen und Computer



Im Blickpunkt

Horst Zuse
60 Jahre Computerentwicklung:
Konrad Zuses Z3 1941-2001

APL-Germany e.V.
Jahrgang 20 Nr. 1 2001
ISSN - 1438-4531
RHOMBOS-VERLAG

Inhaltsverzeichnis

David Liebtag
Using Namespaces
A Practical Example 4

Axel Güth
APL2 für Windows
Verbesserungen im Service Level 8 9

Horst Zuse
60 Jahre Computerentwicklung
Konrad Zuses Z3 1941-2001 10

Conrad Hoesle-Kienzlen
Vom Einzelplatz ins Internet
Die Fallstricke bei der Client/Server-Programmierung 23

Herbert Voß
LYX
Open Source Document Processor 29

Dieter Kilsch
Durchdringung von Kugel und Zylinder
Mit einem Programm die Masse berechnen 44

Wolf Dehus
Opa und der neue Computer 47

Nina Kroll
Frauen und Computer 49

Aufnahmeformular 50

Impressum 2

Editorial 3

4 Using Namespaces: A Practical Example

APL2 namespaces provide a variety of benefits to the application developer. Although mainframe APL2 has supported namespaces for years, they are used less than one might suppose. Many people find namespace concepts confusing, the documentation bewildering, and the prospect of actually using them daunting. This article will attempt to remove the confusion and show that namespaces are really easy to use.

10 Computerentwicklung: Konrad Zuses Z3 1941-2001

Am 12. Mai 2001 jährte sich zum 60. Mal der Tag, an dem Konrad Zuse seine vollautomatische Rechemaschine Z3, unbemerkt von der Öffentlichkeit, einer kleinen Gruppe von Wissenschaftlern vorführte. Die Teilnehmer bekamen eine in allen Komponenten funktionsfähige vollautomatische, programmgesteuerte Rechenmaschine zu sehen, die auf der binären Logik und dem binären Gleitkommazahlensystem basierte. Zu dieser Zeit gab es weltweit keine auch nur annähernd ähnliche Maschine. Die Maschine Z3 kann man als den Prototyp des heutigen modernen Computers bezeichnen.

23 APL-Anwendungen: Client/Server-Programmierung

Die herkömmlichen APL-Anwendungen laufen in der Regel für jeden Benutzer getrennt in einem eigenen Adressraum. Das heißt, für jeden Benutzer wird ein eigenes APL gestartet. Alle Variablen, die den Stand der Anwendung zu einem beliebigen Zeitpunkt widerspiegeln, sind ständig vorhanden und ihr Zustand ändert sich nur auf Grund von Benutzer-Interaktionen. Das Gleiche gilt für den Statusindikator. In einer Client/Server-Umgebung ist das fundamental anders. Hier können mehrere Benutzer das selbe Programm in einem einzigen Adressraum verwenden.

29 Open Source: LyX - der Document Processor

LyX ist eine grafische Oberfläche, mit deren Hilfe man auf der Grundlage des professionellen Satzprogramms LaTeX hochwertige Satzdokumente erzeugen kann, ohne jedoch spezielle Kenntnisse von LaTech besitzen zu müssen. In diesem Beitrag werden wesentliche Eigenschaften dieses Frontends zu TEX vorgestellt (Version LYX 1.1.6fix2). Eine erheblich erweiterte 1.2 Version ist für den Herbst zu erwarten.

Impressum APL - Journal ISSN 1438-4531

Herausgeber: Dipl.-Math. Dieter Lattermann, Walldorf

Redaktion: Dipl.-Volksw. Martin Barghoorn (verantwortlich)
Technische Universität Berlin
Franklinstr. 28
D-10587 Berlin
Tel. (030) 314 24392
Fax (030) 314 25901
Email: barg@cs.tu-berlin.de

Erscheinungsweise: halbjährlich

Erscheinungsort: Berlin

Verlag: RHOMBOS-VERLAG, Berlin
Kurfürstenstr. 17
D-1785 Berlin
Tel. (030) 261 9461
Fax (030) 261 6300
Email: verlag@rhombos.de,
Web: www.rhombos.de

Gestaltung: RHOMBOS-VERLAG, Berlin

Druck: Kopierfabrik, Eberswalde

Copyright: *APL Germany e.V.*
(für alle Beiträge, die als Erstveröffentlichung erscheinen)

Homepage des APL-Journals: http://www.rhombos.de/apljourn.htm
FTP-Server zum Datei-Upload: ftp.cs.tu-berlin.de/pub/lang/apl/stat/incoming.

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung von Redaktion und Herausgeber nicht übernommen werden. Sämtliche Veröffentlichungen im APL-Journal erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Mit Namen gekennzeichnete Artikel geben nicht unbedingt die Meinung des Herausgebers oder der Redaktion wieder. Für unverlangte Einsendungen wird keine Haftung übernommen. Nachdruck ist nur mit Zustimmung des Herausgebers sowie mit Quellenangabe und Einsendung eines Beleges gestattet. Überarbeitungen eingesandter Manuskripte liegen im Ermessen der Redaktion.

Wie immer freuen wir uns auf Ihr Feedback zu diesem Heft. Schreiben Sie uns an barg@cs.tu-berlin.de, oder faxen Sie an 030 - 314 25901.

Technische Hinweise für Autorinnen und Autoren

Manuskripte werden von der Redaktion entgegengenommen. Sie müssen im Besitz der Nutzungsrechte für die von Ihnen eingereichten Texte, Fotos und Zeichnungen sein. Die Zustimmung zum Abdruck wird vorausgesetzt.

Manuskripte und Beiträge aller Art werden in jeder Form entgegengenommen. Es werden jedoch Beiträge in gespeicherter Form bevorzugt. Bitte speichern Sie Ihren Text im Format Ihrer Textverarbeitung und zusätzlich als RTF-Datei (Rich Text Format). Sie brauchen sich nicht die Mühe machen, Ihren Text aufwendig zu formatieren. Auszeichnungen im Text sollten sich auf „Kursiv“ beschränken. Ihre Datei schicken Sie der Redaktion bitte auf einem Datenträger (Diskette oder CD-ROM). Alternativ hierzu können Sie der Redaktion Ihre Daten auch per Email (barg@cs.tu-berlin.de) oder auch zum FTP-Server der Technischen Universität Berlin schicken. Die Adresse lautet: ftp.cs.tu-berlin.de/pub/lang/apl/stat/incoming.

Auch ein komfortables FTP-Transferprogramm für Windows gibt es beim ftp-Server des Fachbereichs Informatik der TUB unter: ftp.cs.tu-berlin.de/pub/lang/apl/bin/

Wenn Sie Ihr Manuskript einreichen, sollten Sie zusätzlich eine Gliederung und Zusammenfassung Ihrer Arbeit vorlegen. In der Zusammenfassung sollte in wenigen Sätzen die wichtigste Aussage Ihres Textes wiedergegeben sein.

Grafiken

Bitte senden Sie uns einen sauberen Ausdruck der Graphiken auf rein-weißem Papier. Linien (keine Haarlinien!) in den Graphiken sollten Schriften so groß gewählt sein, dass sie auch bei (stark) verkleinerter Wiedergabe im Heft noch zu erkennen sind; auf der Rückseite vermerken Sie bitte Ihren Namen und die Nummer der Abbildung; Zusätzlich wäre es schön, wenn Sie uns die mit einem gängigen Vektor-Grafikprogramm erstellten Zeichnungen und Grafiken auch als Datei liefern. Grafikdateien (Strichzeichnungen) sollten möglichst im Austauschformat EPS (skalierbar) mit 300 bis 600 dpi Auflösung in schwarzweiß bzw. Graustufe abgespeichert werden. Nach Rücksprache können sie zusätzlich Ihre Grafiken in Ihrem Original-Dateiformat liefern oder in ihr Textdokument einbetten.

Digitale Fotos benötigen wir im TIF-Format mit einer maximalen Auflösung von 250 dpi. Nicht geeignet sind Abbildungen und Fotos, wenn Sie aus dem Internet stammen beziehungsweise im Original als Bildschirmdarstellung (Auflösung 75 - 150 dpi) vorliegen. Ebenfalls nicht geeignet sind Grafiken, die mit einem pixelorientierten Zeichenprogramm erstellt wurden.

Editorial

Walldorf, 11. Juli 2001

Liebe APL-Freunde,

pünktlich für das Sommerloch ist das Ungeheuer von Loch Ness aufgetaucht oder zumindest ein Krokodil im Rhein. Diese Meldungen werden Sie aber kaum hinter dem Ofen – oder besser aus Ihren kühlen Räumen - hervor locken.

Ich hoffe, dass hingegen ein ebenso regelmäßiges Ereignis wie das Erscheinen des *Journals* auch diesmal wieder Ihr Interesse findet. Wenn ja, verdanken wir das den Autoren der Beiträge und dem unermüdlichen Einsatz unseres Editors. Ihnen sei herzlich gedankt.

Wie Sie dem beigefügten Protokoll der Mitgliederversammlung entnehmen können, die im April in Bingen stattfand, ist unser Vorstand wieder vollständig. Wir haben einen neuen Schriftführer – Conrad Hoesle-Kienzlen – und einen neuen Schatzmeister – Jürgen Beckmann; mit dem stellvertretenden Vorsitzenden Martin Barghoorn und mir müssen Sie weiter leben. Ich denke, der teilerneuerte Vorstand wird wieder ein gutes Team bilden. Wenn nein – die Mitgliederversammlung hat es so gewollt und ist selbst schuld.

Unsere nächste Tagung wird voraussichtlich am 15. und 16. November 2001 in Hamburg als gemeinsame Veranstaltung des GSE-APL-Arbeitskreises und des *APL Germany e.V.* stattfinden. Der zweite Tag wird von *APL Germany* bestritten werden, ich bitte Sie jetzt schon, über mögliche eigene Beiträge und Vortragswünsche nachzudenken.

Unser Verein wird dem GSE-APL Arbeitskreis beitreten, um die auf der letzten Mitgliederversammlung beschlossene engere Zusammenarbeit auf eine offizielle Basis zu stellen. Damit werden Sie die Möglichkeit haben, an GSE-Veranstaltungen teilzunehmen, auch wenn diese nicht gemeinsam mit *APL Germany* abgehalten werden.

Ich wünsche Ihnen Gewinn bringende Lektüre dieses *Journals* und einen schönen Sommer und grüße Sie herzlich,

Ihr

Dieter Lattermann

David Liebtag

Using Namespaces:

A Practical Example

APL2 namespaces provide a variety of benefits to the application developer:

- **Namespaces reduce the chance of name conflicts between applications. Each namespace has its own namescope. By placing objects in a namespace, the objects' names are isolated; they are protected from conflict with names in other namescopes.**
- **Namespaces hide application internals. Users can access top-level functions without being exposed to internal utility functions.**
- **Namespaces make it easier to combine and reuse APL applications. Multiple users and applications can access code without requiring multiple copies of the code.**
- **Namespaces make it easier to maintain multiple user applications. All users can access the same namespace. A single namespace update effects all users.**
- **Namespaces are stored as load modules and so can be managed like programs written in other languages.**

Although mainframe APL2 has supported namespaces for years, they are used less than one might suppose. Many people find namespace concepts confusing, the documentation bewildering, and the prospect of actually using them daunting. This article will attempt to remove the confusion and show that namespaces are really easy to use.

APL2 Symbols

Most applications are easy to manage when they are new. But over the years, they get enhanced and fixed, over and over again. And each time they're changed, a few more functions and variables are added. Eventually, what may once have been a nice small application has grown into a huge workspace. These large workspaces may run into an APL2 system limit: a workspace may only contain 32767 symbols.

The *APL2 Language Reference Manual* defines a symbol as a unique name or constant. This means that a symbol is the name of a variable, a defined function, a defined operator, or a constant like 'Vector' or 3.14159. A symbol may be created in immediate execution or during definition of a function or operator. The number of symbols used can be queried using the)Symbols system command. Even in an empty workspace, the system uses a few symbols:

```
      )CLEAR
CLEAR WS
      )SYMBOLS
Is 50
```

Here's what happens if you define a small program:

```
    ∇FOO;LV
[1] LABEL: 3.45 'ABC' 3J4
[2] ∇
    )SYMBOLS
Is 56
```

One slot in the symbol table is used for each of the names, FOO, LV, and LABEL and each of the constants 3.45, 'ABC', and 3J4.

A typical application workspace can easily use hundreds or even thousands of symbols:

```
      )LOAD 2 GRAPHPAK
      )SYMBOLS
Is 1363
```

Although a single workspace may only contain 32767 symbols, each additional namespaces may contain also 32767 symbols. Therefore, namespaces can be used to combine tools into arbitrarily large applications.

Creating a Namespace

The *Processor 11 – Access To Namespaces* chapter of the *APL2 System Service Reference* manual has a lot of information about how to create many sophisticated types of namespaces, but most of it is unnecessary for average applications. Here is a session log that illustrates everything you need to know to package a workspace into a namespace. It is followed by a step-by-step explanation (see figure 1. page 5):

First, the workspace is saved into a private library. The workspace packaging facility can only access workspaces that are saved in private libraries.

Next, an object dataset is allocated to the ddname SYSPUNCH. The workspace packaging facility produces an object file as an intermediate step in the namespace creation process. The object file is written to the dataset allocated to the ddname SYSPUNCH, so the example uses the TSO command ALLOC to allocate an object dataset to SYSPUNCH.

Note:

To package a workspace, you will need two datasets. They are both normal datasets that you can create yourself using PDF or the TSO ALLOC command. The object dataset should have a logical record length of 80 bytes and should be fixed record format. The final namespace dataset is a load module dataset and should be record format U. If you are unfamiliar with creating datasets, consult the figures in Appendix A for further information.

Once the workspace is saved in a private library and SYSPUNCH is allocated, associate and call the PACKAGE function. The right argument of PACKAGE is simply the name of the workspace to be packaged. The result is the name of the dataset to which the output object file is written.

```
⍝ Step 1: Save the workspace
      )LOAD 2 GRAPHPAK
Saved 2000-12-06 11.59.46 (GMT-8)
LICENSED MATERIALS - PROPERTY OF IBM
5688-228 (C) COPYRIGHT IBM CORP. 1984, 1994.
      )WSID GRAPHPAK
Was 2 GRAPHPAK
      )SAVE
2001-03-19 14.23.25 (GMT-8) GRAPHPAK

⍝ Step 2: Allocate SYSPUNCH
  )HOST ALLOC SHR REUSE FI(SYSPUNCH) DA(SPACES.OBJ(GRAPHPAK))
TSO(0)

⍝ Step 3: Package the workspace
      3 11 ⎕NA ⍞PACKAGE⍞
1
      PACKAGE ⍞GRAPHPAK⍞
LIEBTAG.SPACES.OBJ

⍝ Step 4: Link-edit the object deck into a load module
      )HOST AP2MP11L LIEBTAG.SPACES.LOAD(GRAPHPAK) LIEBTAG.SPACES.OBJ(GRAPHPAK)
TSO(0)
```

Figure 1: Step-by-step explanation

The final step uses AP2MP11L to link-edit the object deck into a load module. AP2MP11L is a Rexx exec that is used to link-edit namespaces and other external modules. AP2MP11L takes two arguments: the name of the dataset into which the load module should be placed and the name of the object dataset.

Notes:

The System Services Reference manual describes several additional link-editing techniques using JCL and other TSO commands, but AP2MP11L is much easier.

AP2MP11L is shipped in the APL2 sample clist libraries. Your system programmer may have made it available for your use. To check if AP2MP11L is available, simply try to issue it with no arguments in a)HOST command. If it is available, you will see instructions about how to use the command. If you do not see these instructions, AP2MP11L is not available and you should ask your system programmer to add it to a dataset allocated to SYSPROC.

Making Associations

To use your namespaces, you need to make them available to APL2. To do this, allocate the load module dataset to a ddname. You can use any ddname you prefer. You normally perform this allocation during logon or in the APL2 invocation clist. Here is an example that uses the TSO command ALLOC to allocate the load module dataset to the ddname SPACES.

```
      )HOST ALLOC SHR REUSE FI(SPACES) DA(SPACES.LOAD)
TSO(0)
```

Finally, you can associate names with objects in the namespace. Notice that when we associate the names PLOT and VIEW, the rest of the symbols in Graphpak do not contribute to the)Symbols result.

```
)CLEAR
CLEAR WS
      'SPACES.GRAPHPAK' 11 ⎕NA⊃'PLOT' 'VIEW'
1 1
      )SYMBOLS
53
      PLOT ?10⍴10
      VIEW
```

When using namespace it is often necessary to associate many names. It is usually a good idea to maintain a group variable in a namespace that contains the list of names that should be associated. This allows the user to simply access the list and associate all the names in the list. For example:

```
'SPACES.GRAPHPAK' 11 ⎕NA 'GPDEMOFNS'

1
      'SPACES.GRAPHPAK' 11 ⎕NA GPDEMOFNS

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

You can also use surrogate names and associate names with system functions in namespaces. You can use this technique to dynamically build lists of objects. For example, here's how you might associate names with all the variables in Graphpak:

```
      'SPACES.GRAPHPAK' 11 ⎕NA 'GPAK_QNL ⎕NL'
1
      'SPACES.GRAPHPAK' 11 ⎕NA GPAK_QNL 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

You can also make associations from namespace namescopes to the active namescope. To make an association to an object in the active namescope, use '' 11 as the name association left argument. For example:

```
      '' 11 ⎕NA 'NAME'
1
```

Namespaces with Side Effects

Some applications have side effects. For example, a program that reads object definitions from a library may establish the definitions in the workspace. These applications may require special modifications when they are packaged. Their side effects typically should occur in the caller's namescope, not the namespace's namescope.

To continue the example, a program that establishes objects should probably establish the objects in the caller's namescope. The code that establishes the objects may need to be enhanced to reach across the namescope boundary.

It is a good idea to design applications so they can run either packaged as a namespace or unpackaged in the active namescope. If an application is written so it can only be run in a pakkaged state, then it must be packaged to test changes. Debugging and developing enhancements is usually much easier if the application can be run in the active namescope.

The QNS external function can be used to determine whether the current namescope is the active namescope or is a namespace's namescope. By judicious use of QNS, you can write applications that will run either packaged or unpackaged and are easy to maintain.

Here is a simple example application that has side effects and will run either in the active namescope or packaged in a namespace. The function uses QNS to determine whether it is packaged. If it is not, it builds a cover function for ⎕FX in the current namescope. If it is packaged, it associates a name with ⎕FX in the caller's namescope. Notice that once the name QFX is established, it can be used in the body of application without regard to whether it is in the same namescope or not:

```
  ∇Z←FIX RA;QNS;QFX
[1] ⎕ES(1≠3 11 ⎕NA 'QNS')/'QNS UNAVAILABLE⍕
[2] →('' 11≡QNS 0)/ACTIVE ⍝ QNS returns '' 11 in the active namescope
[3]
[4] ⍝ Running in packaged namescope, establish association to caller's namescope
[5] ⎕ES(1≠'' 11 ⎕NA 'QFX ⎕FX')/'⎕FX UNAVAILABLE'
[6] →BODY
[7]
[8] ACTIVE: ⍝ Running in active namescope, fix a cover function
[9] ⎕ES(~'QFX'≡⎕FX 'Z←QFX RA' 'Z←⎕FX RA')/'QFX UNAVAILABLE'
[10]
[11] BODY:
[12] Z←QFX RA
[13] ∇
```

Namespace Architecture

Applications typically are deployed as one or more workspaces. The workspaces may contain groups of tools to perform tasks such as file IO, database access, user interface, calculations, and report generation. These groups of tools may exist in several different applications' workspaces. They can be more easily managed if they are partitioned into separate namespaces. For example, the file IO tools might be placed in a single namespace. Then, all the applications that need to perform file IO can associate names with the tools in the file IO namespace.

However, splitting a large application into namespaces can be difficult. Determining which objects should be placed in each namespace may not be at all obvious. Although it may seem that the application's group variables are a good starting point, this is often not the case. Group variables are usually designed so that users can copy a working subset of an application. This is not the goal when trying to partition an application into namespaces. Rather, you want to isolate working sets of tools. A good rule of thumb is that each set of tools should have its own namespace. Expert knowledge of the sets of tools used by an application is

often essential to partition it into namespaces.

Determining the control flow within an application may also be important. Some applications create semi-global variables that are used throughout the application. Wherever possible, semi-global variables should be removed. They should be replaced with code that uses arguments to pass data between functions. However, this is not always possible or desirable. In these cases, once an application is separated into namespaces, several namespaces may require access to semi-global variables. Because semi-global variables are created at runtime, new initialization code may have to be written to establish associations from the namespaces to these semi-global variables.

Summary

Namespaces are a good way to deploy both large and small applications. They provide namescope isolation, hide application internals, enhance code reusability, and ease maintenance of multiple user applications. Without the assistance of system programmers, developers can package both groups of tools and complete applications and share them with other developers or end users.

Creating namespaces is easy. Using just the information in this article, you can package workspaces yourself. We suggest you try it. You'll probably decide you like it!

Appendix A

PDF 3.2 object dataset allocation panel:

```
                              Allocate New Data Set
Command ===>

Data Set Name  . . . : LIEBTAG.SPACES.OBJECT

Management class . . .                (Blank for default management class)
Storage class  . . . .                (Blank for default storage class)
 Volume serial . . . .                (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . TRACK          (BLKS, TRKS, CYLS, KB, MB, BYTES or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 100            (In above units)
 Secondary quantity    100            (In above units)
 Directory blocks  . . 100            (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . . 32720
 Data set name type  : PDS            (LIBRARY, HFS, PDS, or blank)  *
                                      (YY/MM/DD, YYYY/MM/DD
 Expiration date . . .                 YY.DDD, YYYY.DDD in Julian form
Enter "/" to select option            DDDD for retention period in days
   Allocate Multiple Volumes           or blank)
```

** Specifying LIBRARY may override zero directory block)*

*** Only one of these fields may be specified)*

PDF 3.2 load module dataset allocation panel:

```
                              Allocate New Data Set
Command ===>

Data Set Name  . . . : LIEBTAG.SPACES.LOAD

Management class . . .                (Blank for default management class)
Storage class  . . . .                (Blank for default storage class)
 Volume serial . . . .                (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . TRACK          (BLKS, TRKS, CYLS, KB, MB, BYTES or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 100            (In above units)
 Secondary quantity    100            (In above units)
 Directory blocks  . . 100            (Zero for sequential data set) *
 Record format . . . . U
 Record length . . . . 0
 Block size  . . . . . 4096
 Data set name type  : PDS            (LIBRARY, HFS, PDS, or blank)  *
                                      (YY/MM/DD, YYYY/MM/DD
 Expiration date . . .                 YY.DDD, YYYY.DDD in Julian form
Enter "/" to select option            DDDD for retention period in days
   Allocate Multiple Volumes           or blank)


* Specifying LIBRARY may override zero directory block)

** Only one of these fields may be specified)
```

Contact:

David Liebtag, APL Products and Services, IBM Silicon Valley Laboratory, San Jose, California, eMail: liebtag@us.ibm.com

Der vorstehende Beitrag von David Liebtag (Chefentwickler von APL2) bildete die Grundlage für einen Vortrag, den der Autor in leicht überarbeiteter Form im Juni auf der APL 2001 - Konferenz in New Haven, CT, USA gehalten hat.

APL2000
2001 User Conference Information

http://www.apl2000.com

Save The Date: Sunday, November 4, 2001 - Wednesday, November 7, 2001

Location:
Naples Beach Hotel and Golf Club
851 Gulf Shore Boulevard North
Naples, Florida 34102 USA
www.NaplesBeachHotel.com

Spectacular sunsets over the Gulf of Mexico, white sandy beaches, rolling green golf course with adjacent tennis center and APL - a winning combination

Plan to join fellow APL enthusiasts at our Sixth Annual APL2000 User Conference to be held Sunday, November 4, 2001 through Wednesday, November 7, 2001 at the Naples Beach Hotel and Golf Club in Naples, Florida

The closest airports are Southwest Florida International Airport in Ft. Meyers, Florida and Naples Airport.

Look for upcoming registration information. For further information contact Sonia Beekman at 301-564-9020 or sonia@apl2000.com

Axel Güth

APL2 für Windows

Verbesserungen im Service Level 8

Zusätzlich zu den Neuerungen des Service Level 7, die im APL Journal 2/2000 beschrieben wurden, sind in dem Anfang Juni 2001 erschienen Service Level 8 weitere Verbesserungen enthalten.

AP 145

- Eine neue Eigenschaft „TOP INDEX“ wurde für die Dialogelemente „LISTBOX“, „MLE“ und „LISTVIEW“ eingeführt.

- Die Eigenschaft “SELECTION” wurde auf die Dialogelemente “MLE” und “ENTRY FIELDS” ausgedehnt.

Session Manager und Object Editor

- Ein neues Objektmenü wurde beim Session Manager und beim Object Editor hinzugefügt. Es erscheint auf Drücken der rechten Maustaste. Wenn die Maus auf einen Objektnamen zeigt, kann man für das betreffende Objekt Informationen anzeigen lassen (siehe Abbildung 1), den Editor dafür öffnen, es in die Zwischenablage kopieren, und, wenn man im Editor ist, es in der Kopfzeile der Funktion lokalisieren. Man kann vom neuen Menü aus ebenso die Zwischenablage abfragen und den dortigen Inhalt an der gewünschten Stelle einfügen.

```
Name:        DEMO
Class:       Function
Arguments:   0
Result:      No
Fix time:    23.05.95 20:34:32
Execution properties:
   Can be displayed:        Yes
   Can be suspended:        Yes
   Ignores attentions:      No
   Reflects Domain Error:   No
```

Abbildung 1: “Info” zu Funktion “Demo”

- In der Werkzeugleiste wurden Knöpfe für die Funktionen “Suppress” und “Attention” des Menüs “Signals” hinzugefügt (Abbildung 2).

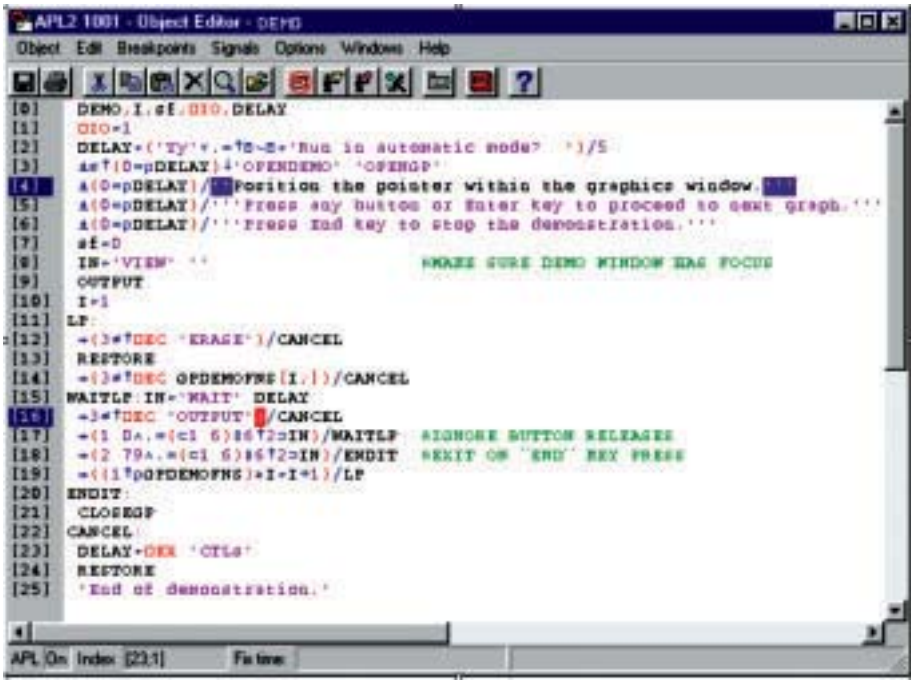


Abbildung 2: Anzeige unpaariger Hochkommata und Klammern.

- Neue Systemoptionen zur Definition von Standard-Schriftarten in den Session-Manager- und Object-Editor-Fenstern sind jetzt verfügbar. Dafür ist auch eine Checkbox “Use default font” im “Font”-Dialog vorhanden.

- Die neue Einrichtung zur Formatierung von Funktionen und Operatoren und zu deren Prüfung auf doppelte Labels kann über eine neue Option “Format and Check” im “Edit”-Menü oder über F5 jederzeit während der Editor-Sitzung aufgerufen werden. Funktionen und Operatoren werden automatisch formatiert, wenn der Editor erstmalig geöffnet wird.

- Unpaarige runde und eckige Klammern und Hochkommata werden jetzt direkt beim Editieren in Reverse Video angezeigt (siehe Beispiel in Abbildung 2).

Alle Neuerungen sind in den entsprechenden Online-Hilfen sowie im Online User’s Guide ausführlich beschrieben.

Sonstiges

In Service Level 8 sind Fehlerbehebungen für 46 gemeldete Problemfälle in APL2/Win enthalten.

Dieser Service Level ist wie alle seine Vorgänger kumulativ, d.h. er enthält auch alle voran gegangenen Verbesserungen und Fehlerbehebungen. Wenn also ein APL2-Benutzer mal vorher gehende Service Levels nicht installiert hat, so kann er alles auf einmal mit der Installation von Service Level 8 nachholen. Auch für APL2 für Windows Runtime Environment (RTE) ist ein neuer Service Level erschienen. Anwendungen, die auf APL2/Win 1.08 erstellt oder geändert wurden, laufen auf RTE nur mit Service Level 8.

Service Level 8 für APL2 für Windows und für RTE kann herunter geladen werden von www.apl-online. de, dem APL-Internet-Portal der Dittrich & Partner Consulting GmbH.

Kontakt:

Axel Güth, AGueth@compuserve.com

Horst Zuse

60 Jahre Computerentwicklung

Konrad Zuses Z3 1941-2001

Am 12. Mai 2001 jährte sich zum 60. Mal der Tag, an dem Konrad Zuse seine vollautomatische Rechemaschine Z3, unbemerkt von der Öffentlichkeit, einer kleinen Gruppe von Wissenschaftlern vorführte. Die Teilnehmer A. Teichmann, K. Koch, C. Schmieden, H. Schreyer und P. Mayer bekamen eine in allen Komponenten funktionsfähige vollautomatische, programmgesteuerte Rechenmaschine zu sehen, die auf der binären Logik und dem binären Gleitkommazahlensystem basierte. Heutzutage ist nachgewiesen, daß es zu dieser Zeit weltweit keine auch nur annähernd ähnliche Maschine gab. Mit der Maschine Z3, die man als den Prototyp des heutigen modernen Computers bezeichnen kann, leitete Konrad Zuse das Computerzeitalter ein.

Der vorliegende Bericht beschreibt aus Anlaß des 60. Jahrestages Konrad Zuses Rechenmaschine Z3.

Konrad Zuse (1910-1995), der von 1928-1934 an der Technische Hochschule Charlottenburg studierte, wird heute fast einhellig auf der ganzen Welt als Schöpfer des ersten programmgesteuerten und frei programmierbaren Rechners in binärer Gleitpunktrechnung, der wirklich funktionierte, anerkannt. Diese Maschine - Z3 genannt - vollendete er 1941 in seiner kleinen Werkstatt in der Methfesselstraße 7 (Siehe Bild) zu Berlin-Kreuzberg. Erste Gedanken Zuses über die logischen wie technischen Prinzipien zum Bau solcher - völlig neuartiger - Rechnersysteme, die wir heute Computer nennen, gehen bereits auf das Jahr 1934 zurück. Konrad Zuse schuf auch - was weit weniger bekannt ist - mit dem Plankalkül (Entwicklung von 1942-1945 mit der Endfassung 1945 in Hinterstein/ Allgäu) /ZUSE45/ die erste höhere Programmiersprache der Welt. Damit schuf er schon 1945/46 die Symbiose zwischen Hardware und Software.

In der Vergangenheit haben Wissenschaftler wie Ingenieure lange Debatten darüber geführt, welche Komponenten einen Computer ausmachen und wer als wahrer Erfinder anzuerkennen ist. Auf der Weltmathematiker-Konferenz 1998 fand in Paderborn der Kongreß *International Conference on History of Computing* vom 14.8.-16.8.98 statt. Hier trafen renommierte Experten aus aller Welt zusammen. Auf der abschließenden Podiumsdiskussion: *Who invented the Computer?,* sprachen die Fachleute mit überwältigender Mehrheit Konrad Zuse die größte Bewunderung für seine Leistungen auf dem Gebiet der Computerentwicklung aus. Der Artikel ist wie folgt strukturiert: Im Kapitel 3 stellen wir die Architektur der Maschine Z3 dar, in Kapitel 4 erläutern wir den Nachbau der Maschine Z3 aus dem Jahr 2001, im Kapitel 5 geben wir einige abschließende Bemerkungen und in Kapitel präsentieren wir die verwendete Literatur.

Konrad Zuse (1935)

Ehemalige Werkstatt in der Methfesselstraße 7 zu Berlin-Kreuzberg

1 Konrad Zuses Rechner

Die erste Frage, die Zuse sich etwa 1934 stellte, lautete: Welche mathematischen Probleme soll eine Rechenmaschine lösen? Seine Antwort bestand in der folgenden Definition des Rechnens (1936): *Aus gegebenen Angaben(1) nach einer Vorschrift neue Angaben bilden.* Im Jahr 1943 erweiterte / verallgemeinerte er den Begriff des Rechnens und schrieb: *Rechnen ist die Ableitung von Resultatangaben aus irgendwelchen Angaben nach einer Vorschrift.* Obwohl Konrad Zuse von 1936 bis 1945 die vier Rechenmaschinen Z1-Z4 baute, konstruierte er in Wirklich-

1 Konrad Zuse verwendete des öfteren den Begriff: *Angaben.* In einem Bericht von 1937 /ZUSE37a/ beschreibt Konrad Zuse sehr detailliert, was er unter Angaben versteht: *Diese können sehr verschiedene Bedeutungen haben, z.B. Zahlen, Aussagen, Namen, Kennziffern, Dienstgrade, Daten, Befehle, Nachrichten, Schlußfolgerungen, usw.* Er schreibt dann weiter: *Gemeinsam ist allen die Variabilität ihrer Aussage, denn wenn jede Ausgangsangabe nur eine Möglichkeit zuließe, so wäre das Rechnen überflüssig, da ja dann nur noch ein Resultat in Frage käme. So ist z.B. das Vorzeichen einer Zahl zweifach variabel, eine Dezimalzahl 10-fach, ein Buchstabe 26-fach, die Angabe über die Batallionszugehörigkeit dreifach variabel.*

keit nur eine Maschine, denn die Maschinen Z1, Z3 und Z4 verfügen über die gleiche Architektur und unterscheiden sich nur in der zahl der Mikrosequenzen und der Wortlänge. Die Maschine Z2 war ein Versuchsmodell mit einem 16-Bit Festkommarechenwerk zur Erprobung von Relaisschaltungen.

1.1 Konrad Zuses Vision von einer vollautomatischen Rechenmaschine

Konrad Zuses Vision war, den Ingenieuren die stupide Arbeit des Rechnens abzunehmen. Am 30.1.1936 schreibt Konrad Zuse /ZUSE36a/ dazu: *Die Anforderungen, die an die ideale Rechenmaschine des Ingenieurs gestellt werden müssen, gehen aus den folgenden Überlegungen hervor:*

- *Der Ingenieur hat viel mit festen Formeln zu arbeiten, die immer wiederkehren. Man hat gewisse Ausgangswerte, und die Arbeit besteht nun darin, durch bestimmte, für eine Formel immer gleiche Aufeinanderfolge von Grundrechenarten zwischen bestimmten Zahlen das Resultat zu berechnen.*

- *Der Ingenieur braucht Rechenmaschinen, die diese Rechenoperationen automatisch ausführen, indem der Rechenplan auf einem Lochstreifen festgehalten wird, der die Befehle für die einzelnen Rechenoperationen selbsttätig und nacheinander an die Maschine gibt.*

- *Die Maschine muß auf Befehl des Lochstreifens jede verlangte Grundrechnung vollautomatisch ausführen. Ferner muß die Maschine über ein Speicherwerk verfügen, in welchem die während der Rechnung auftretenden Zahlen der Nummer nach geordnet werden können, und aus denen durch ein mechanisches Wählwerk jede gewünschte Zahl abgelesen werden kann. Das Speichern und Ablesen der Zahlen wird ebenfalls durch den Befehlslochstreifen dirigiert.*

Bevor Konrad Zuse mit der Konstruktion des ersten Versuchsmodells Z1 im Jahr 1936 begann, machte sich Gedanken über das zu verwendende Zahlensystem für seine Rechenmaschine und führt dazu /ZUSE36/, /ZUSE36a/ aus: *Ist das Dezimalsystem unpraktisch, so kann man ein geeigneteres wählen, vorausgesetzt, daß es keine Schwierigkeiten macht, die Anfangs- und Endwerte der Rechnung ins Dezimalsystem zu übersetzen.* Konrad Zuse diskutiert verschiedene Zahlensysteme, wie z.B. das von Leonardo da Vinci vorgeschlagene Duodezimalsystem mit der Grundzahl 12, oder die Zahlensysteme zur Basis 8 und 16. Er führt dann zu den Vor- und Nachteilen dieser Zahlensysteme aus: *Von diesen Vorstellungen* (Autor: Welches Zahlensystem hat Vorteile oder Nachteile) *muß man sich aber ganz freimachen, wenn man das für die Maschine beste Zahlensystem finden will. Für die Maschine kommt es nicht darauf an, Rechnungen im Kopf ausführen zu können, oder Zahlen leicht überblicken zu können. Was das maschinelle Rechnen vom menschlichen Rechnen unterscheidet, ist, daß die Maschine gleichzeitig eine große Zahl von Einzeloperationen parallel ausführen kann. Ist es möglich, Rechenmaschinen dieser Art zu bauen, so ist die Art der damit zu rechnenden Aufgaben lediglich durch die von der Maschine ausführbaren Grundrechnungsarten und der Umfang lediglich durch die Größe des Speichers begrenzt. Beim System mit der kleineren Basis sind also zwar auf Grund der höheren Stellenzahl mehr Einzelmultiplikationen auszuführen, dafür sind diese aber in sich einfacher, da die Ziffernwerte kleiner sind. Das ist für die Maschine wesentlich, denn diese ist im Gegensatz zum Menschen besser im Stande, viele einfache Operationen auszuführen, als wenig komplizierte.*

Das System mit der kleinst-möglichen Basis ist für die Maschine am besten geeignet. Die kleinst mögliche Basis ist 2.

Konrad Zuse schreibt 1936 auch, daß durch das System zur Basis 2 bei der maschinellen Festhaltung auf Lochkarten eine hohe Papiersparnis möglich ist und führt zu Speicherwerken aus: *... sondern noch mehr in der Konstruktion der bei dieser Maschine sehr wichtigen Speicherwerke, welche aus wesentlich einfacheren Elementen hergestellt werden können als bisher (Keine Ziffernräder mit Zahnrädern usw.). Erst so sind Massenspeicherwerke möglich.*

In /ZUSE39/ formuliert Konrad Zuse die Aufgabe seiner neuartigen Rechenmaschine in Bezug auf die Wissenschaftler und Ingenieure (Personen die in großen Sälen Rechnungen durchführten, wurden Rechenknechte genannt:

- *Insbesondere dem Ingenieur, Wissenschaftler und Forscher schematische Rechenarbeit abzunehmen und somit hochwertige Arbeitskräfte für produktive Arbeiten freizumachen.*

- *Neue Berechnungsmethoden zur Lösung technischer Probleme zu ermöglichen und somit dem technischen Fortschritt auf den verschiedensten Gebieten zu fördern. Beispielsweise Qualitätserhöhung der Flugzeuge, Materialersparnis im Bauwesen.*

- *Erschließung von Gebieten, die bisher Rechnungen nicht zugänglich waren und somit Vertiefung der wissenschaftlichen Erkenntnis und Ersparung kostspieliger Versuche.*

Aus diesen Definitionen und Überlegungen leitete Konrad Zuse die Architektur seiner Rechner Z1, Z2, Z3 und Z4 ab. Seine Rechner sollten frei programmierbar sein, d.h. beliebig viele Instruktionen von einem Lochstreifen einlesen und verarbeiten können; sie sollten das binäre Zahlensystem (siehe Abbildung 3) verwenden, da Konrad Zuse seine Rechner aus bistabilen Bauelementen aufbauen wollte. Nicht

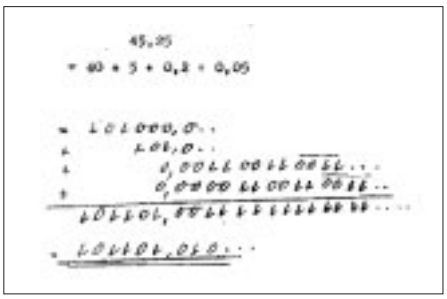


Abbildung 3: Instruktionen auf der Grundlage des binären Zahlensystems (Handskizze von Konrad Zuse)

nur die Zahlen wollte er binär darstellen, auch die gesamte Maschine sollte auf dem 0-1 Prinzip (Aussagenlogik) arbeiten. Er entwickelte ein leistungsfähiges binäres Gleitkommarechenwerk mit der halblogarithmischen Zahlendarstellung, welches erlaubte, sehr große und sehr kleine Zahlen mit hin-

reichender Genauigkeit zu verarbeiten. Er implementierte dazu ein leistungsfähiges Addierwerk mit dem einschrittigen Übertrag und einer arithmetischen Ausnahmebehandlung (Behandlung von Sonderwerten wie z.B. Unendlich und Null). Er konstruierte einen Speicher zur Speicherung beliebiger Daten, entwarf eine Steuereinheit zur Steuerung des Rechners per Lochstreifen und implementierte Ein- bzw. Ausgabeeinheiten im Dezimalsystem.

1.2 Die Rechenmaschine Z3

Die vollautomatische Rechenmaschine Z3 wird fast einhellig auf der Welt als der erste programmgesteuerte, im binären Gleitkommasystem, arbeitende Rechner, der wirklich funktionierte, anerkannt. Die Maschine Z3 war am 12. Mai 1941 in allen Komponenten funktionsfähig. Wir beschreiben nun die Komponenten der Maschine Z3.

1.2.1 Komponenten der Maschine Z3

Die Maschine Z3 besteht aus den folgenden Komponenten (vergleiche Abbildung 4):

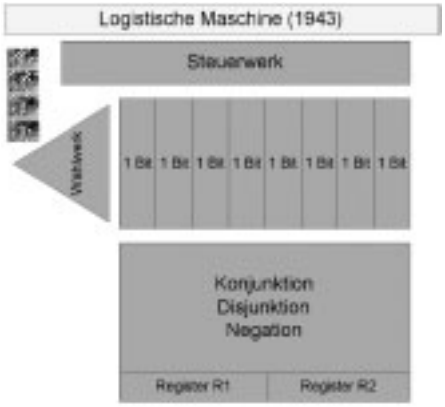


Abbildung 4: *Der Nachbau der Maschine Z3 von 1961 mit den Komponenten: Speicher, Wählwerk, Rechenwerk, Steuerwerk, Eingabe- und Ausgabeeinheit, Lochstreifenleser, Mikrosequenzer und dem Impulsgeber mit der Taktsteuerung. Die Original-Z3 Maschine enthielt noch einen zweiten Speicherblock, während der Nachbau nur einen Speicherblock mit 32 Worten à 22 Bits enthält.*

Die Originalrechenmaschine Z3 wurde im Dezember 1943 in Berlin-Kreuzberg in der Methfesselstraße 7 durch einen Bombenangriff zerstört. Der Nachbau der Maschine Z3 wurde 1961 durch die Zuse KG in Bad Hersfeld mit dem Ziel durchgeführt, die Funktionsfähigkeit der Maschine Z3 zu demonstrieren. Alle folgenden Bilder beziehen sich auf diesen Nachbau der Maschine Z3.

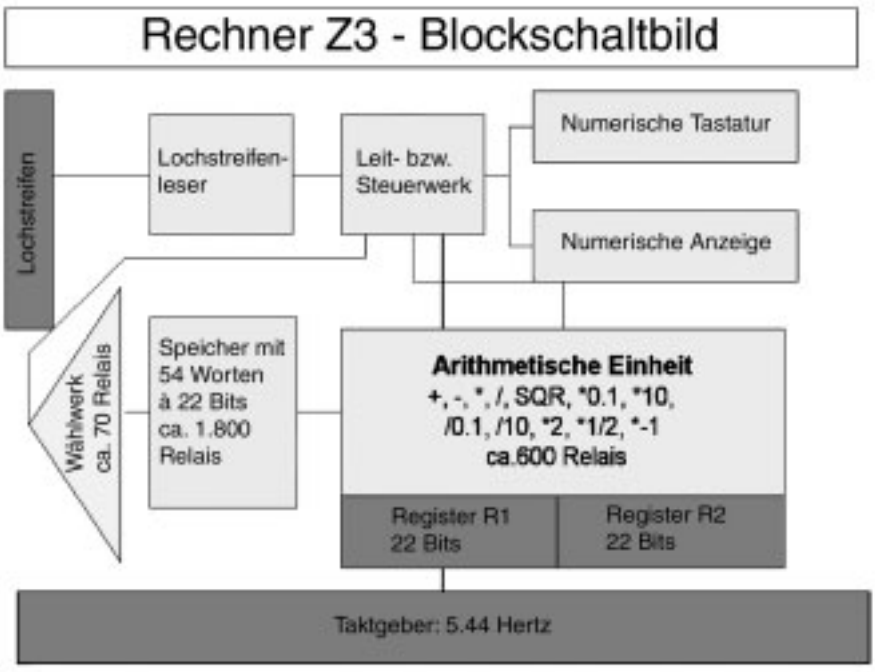


Abbildung 5: *Architektur der Maschine Z3 (1941). Die Maschine Z3 ist eine getaktete Maschine und besteht aus ca. 2400 Relais und neun Schrittschaltern. Die Z3 verfügt über einen Lochstreifenleser (35 mm Kino-Normalfilm) zur Programmsteuerung, einem Steuerwerk / Leitwerk zur Interpretation der Befehle und zur Steuerung der gesamten Maschine, einem Speicher mit 32 Worten a 22 Bits, einer arithmetischen Einheit, den Ein- / Ausgabeeinheiten im dezimalen Gleitkommasystem und einem Taktgeber von ca. 5.3 Hertz. Die Komponenten sind untereinander durch einen parallelen Datenbus verbunden.*

Die Maschine Z3 ist eine getaktete Maschine und besteht bis auf wenige Ausnahmen aus ca. 2400 Telefonrelais und neun Schrittschaltern. Der Takt für die Maschine wird von einem Impulsgeber erzeugt. Jeder Takt versetzt die Maschine von einem Zustand in den nächsten Zustand. Moderne PENTIUM-Prozessoren haben einen Takt von mehr als 700 Millionen Hertz (700 MHZ). Die Maschine Z3 verfügt über einen Takt von ca. 5,3 Hertz. Die gesamte Maschine wird durch das Steuer- bzw. Leitwerk gesteuert. Die Steuerung umfaßt das Einlesen des 35mm Kino-Normalfilms mit den Instruktionen bzw. Befehlen (Rechenplan) an die Komponenten der Maschine, aber auch an das Rechenwerk und den Speicher. Der Lochstreifenleser kann einen 35 mm Kino-Normalfilm einlesen, auf dem die Befehle in einem 8-Bit Kode eingestanzt sind. Der Lochstreifen enthält den Rechenplan (Programm) der Maschine Z3 und ermöglicht die Speicherung beliebig langer Befehlsfolgen. Der Speicher enthielt in der Original Z3 eine Kapazität von 64 Worten a 22 Bis. Der Nachbau der Z3 enthält 32 Worte a 22 Bits. Die Gesamtkapazität des Speichers ist also: 64 * 22 = 1408 Bits. Jedes Wort im Speicher ist durch eine Adresse z mit $1 \leq z \leq 64$ ansprechbar, welche durch das Wählwerk aus der Lochkombination auf dem 35 mm Kino-Normalfilm ermittelt wird. Die arithmetische Einheit ist das Rechenwerk der Maschine. Es ist ein binäres Rechenwerk, welches auf der binären halblogarithmischen Zahlendarstellung (binäre Gleitkommadarstellung) basiert. Die möglichen arithmetischen (Rechenoperationen) Operationen sind: +, -, *, /, SQRT(x).

Die Binärzahlen oder Operanden für eine arithmetische Operation werden in den Registern R1 und R2 abgelegt. Register sind Kurzspeicher. Konrad Zuse nannte sie Bereitschaftsspeicher. Jedes der Register hat eine Kapazität von 22 Bits zur Aufnahme einer binären Gleitkommazahl. Eine Addition

wird z.B. wie folgt vorgenommen. Die binär kodierten Gleitkommazahlen werden aus dem Speicher oder von der Eingabeeinheit in die Register R1 und R2 transportiert. Die Eingabe der Dezimalzahlen erfolgt auf der Eingabeeinheit, nach Abschluß der Eingabe wird die Dezimalzahl in eine binäre Gleitkommazahl konvertiert und im Register R1 oder R2 gespeichert. Die Inhalte der Register R1 und R2 sind die Operanden für die arithmetische Operation der Addition. Addiert wird, indem die Summe von R1 und R2 gebildet werden und das Ergebnis nach R1 zurückgeschrieben wird. Das Register R2 wird auf leer gesetzt. Wir schreiben dies als R1 := R1 + R2, wobei das Zeichen := ein Zuweisungszeichen ist: Die Inhalte von R1 und R2 werden addiert und das Ergebnis wird an R1 zugewiesen.

Die Komponenten arithmetische Einheit, Ein- und Ausgabeeinheiten und der Speicher sind durch einen parallelen Datenbus verbunden, der aus 22 Leitungen (für jedes Bit eine Leitung) besteht. Die Ein- und Ausgabeeinheiten sind für die Kommunikation zwischen dem Menschen und der Maschine vorgesehen. Über die Eingabekonsole werden die Dezimalzahlen für die Rechnung eingegeben und in binäre Gleitkommazahlen konvertiert. Die Ausgabeeinheit zeigt das Ergebnis einer Rechnung, es steht immer in Register R1, als Dezimalzahl an.

Kurzum, die Maschine Z3 ist ein mit binären Gleitkommazahlen, parallel arbeitender programmgesteuerter Digitalrechner.

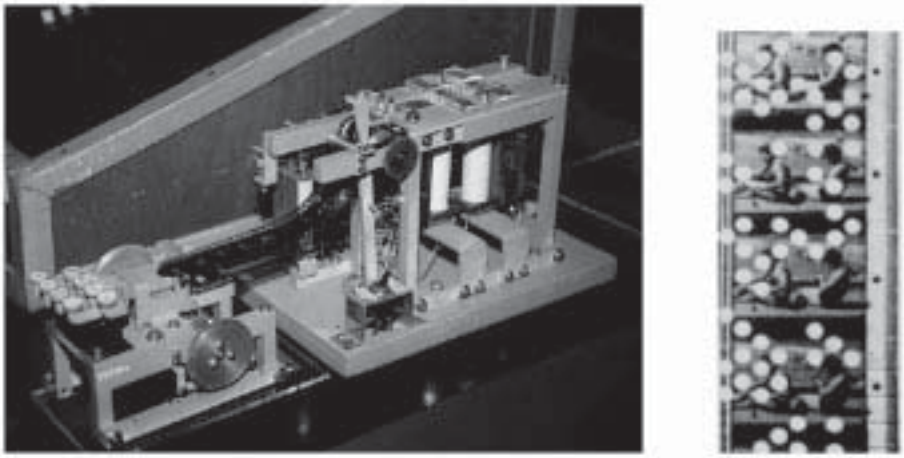


Abbildung 7: *Links: Lochstreifenleser der Maschine Z3 für den 35mm Kino-Normalfilm. Rechts: Der 35mm Kino-Normalfilm auf dem die Anweisungen bzw. Befehle an die Maschine im 8-Bit-Kode gelocht sind.*

1.2.2 Die Rechenplansteuerung der Maschine Z3

Die zentrale Einrichtung der Maschine Z3 ist die Rechenplansteuerung (Programmsteuerung). Konrad Zuse schreibt dazu in der Patentschrift Z23139 vom 11.4.1936 /ZUSE36/: *Vorliegende Erfindung dient dem Zweck, häufig wiederkehrende Rechnungen beliebiger Länge und beliebigen Aufbaues, die sich aus elementaren Rechenoperationen zusammensetzen, mit Hilfe von Rechenmaschinen selbsttätig durchzuführen.*

Voraussetzung für jede Art der auszuführenden Rechnung ist die Aufstellung eines Rechenplanes, in dem die aufeinanderfolgenden Rechenoperationen dem Charakter und der Reihe nach aufgezeichnet werden, und die im Verlauf der Rechnung auftretenden Zahlen fortlaufend numeriert oder nach einem anderen Schema geordnet werden, ohne sie zunächst der Größe nach zu bestimmen. Man geht von bestimmten ,,Ausgangswerten„ aus, die den Variabeln einer Formel entsprechen, und leitet aus diesen durch bestimmte Operationen über eine Reihe von Zwischenwerten die Resultatwerte ab. Ist für eine bestimmte Aufgabe ein solcher Rechenplan einmal aufgestellt, so gilt er für sämtliche Variationen der Ausgangswerte.

Die grundlegende Idee einer vollautomatischen Rechenmaschine ist die Steuerung der gesamten Maschine durch einen Rechenplan, der die Befehle an die Maschine enthält und der beliebig lang sein kann. Eine Maschine, die eine beliebig sinnvolle Sequenz von Befehlen, auch Instruktionen genannt, verarbeiten kann, ist eine frei programmierbare Maschine. Als Medium für den Rechenplan dient ein 35mm Kino-Normalfilm. Auf dem Normalfilm sind die Anweisungen bzw. Befehle an die Maschine Z3 mit acht Lochkombinationen eingestanzt. Jedes Loch im Film ist ein Ja-Nein bzw. ein Bit mit den Zuständen 0 oder 1. Die Befehle sind also in einem acht Bit Kode kodiert. Der 35 mm Normalfilm wird von einem Lochstreifenleser, welcher von der Steuereinheit der Maschine Z3 angesteuert wird, schrittweise eingelesen.

Die Verwendung des 35mm Kino-Film ist für das schrittweise Lesen der Befehle besonders geeignet, da die Perforation des Films zum schrittweisen Transport genutzt werden kann. Das Steuer- bzw. Leitwerk interpretiert

Befehle zur Steuerung der Ein- und Ausgabe

Lu	Eingabe von Dezimalzahlen auf der Eingabeeinheit	
Ld	Ausgabe der Ergebnisse auf der Ausgabeeinheit	

Befehle mit einem Zugriff auf den Speicher

Pr z	Lesen von Adresse z aus dem Speicher nach Register R1 bzw. R2	
Ps z	Speichern in Adresse z des Speichers vom Register R1	

Befehle für die arithmetische Operationen

Ls1	Addition der Register R1 und R2 zu R1:	R1:= R1 + R2
Ls2	Subtraktion der Register R1 und R2 zu R1:	R1:= R1 - R2
Lm	Multiplikation der Register R1 und R2 zu R1:	R1:= R1 * R2
Li	Division der Register R1 und R2 zu R1:	R1:= R1 / R2
Lw	Quadratwurzel aus Register R1 nach R1:	R1:= Wurzel (R1)

Tabelle 1: *Der Rechner Z3 kennt neun Befehle, die über den Lochstreifen kodiert werden können. Mit R1 und R2 sind die Register R1 und R2 bezeichnet. Dies sind Speicher von einem Wort mit einer Länge von 22 Bits. Das Zeichen := ist ein Zuweisungszeichen.*

die Befehle auf dem Lochstreifen und aktiviert bzw. startet die entsprechenden Operationen in der Maschine. Das Steuerwerk der Z3 kann neun verschiedene Befehle interpretieren. Die Befehle lassen sich in drei Klassen einteilen (siehe Tabelle 1).

1.2.4 Speicher der Maschine Z3

Wir wenden uns nun der Architektur des Speichers der Maschine Z3 zu, der im Gegensatz zu dem Speicher der Maschine Z1 vollständig aus Telefonrelais aufgebaut ist.

Konrad Zuse wollte im Speicher aber nicht nur Zahlen ablegen, sondern seine Vorstellungen eines Speichers gingen erheblich weiter /ZUSE37/, Seite 2: B*ei der vorliegenden Erfindung werden diese Schaltglieder zu einem Speicherwerk zusammengesetzt, mit dem beliebige Angaben, z.B. Zahlen, gespeichert werden können. Solche Vorrichtungen spielen besonders bei Rechenmaschinen eine Rolle. Sie können aber auch zur Speicherung anderer Angaben, z.B. von Befehlen an Arbeitsmaschinen (Schaltspeicherung). von Buchstabenkombinationen (z.B. Telegrammspeicherung), von Buchstabenverschlüsselungen (Chiffriermaschinen) od. dgl. benutzt werden.*

Die Angaben, wie z.B. Zahlen, stellte Konrad Zuse als binäre Kodierungen dar, d.h. jede Angabe wurde als eine Folge von 0 und 1 Kombinationen dargestellt. Der weit verbreite ASCII-Code kodiert z.B. Buchstaben in 7-Bitkombinationen. Diese Bitkombinationen können ebenfalls in dem Speicher der Z3 abgelegt werden.

Zu dem Bild rechts in Abbildung 7 finden wir in /ZUSE36/: ..... *zeigt eine elektrische Ausführung des Speicherwerks. Die senkrecht übereinanderliegenden Relais gehören zu einer Zelle, die zur Speicherung einer Zahl dient. Die nebeneinanderliegenden Relais, ...entsprechen den Stellen 1, 2, 3 ...der zu speichernden Zahl. Ferner sind jeder Zelle drei Relais B, C, D zugeordnet. Die Relais A dienen der eigentlichen Speicherung und entsprechen im geschlossenen Zustand der Ziffer 1 und im offenen der Ziffer 0 (bzw. +, -). Über jedem Magneten liegen zwei Anker a und b. Die Negativpole der Magneten sind als Kreise angedeutet. Die Spulen der Magnete A können von zwei Seiten Strom erhalten: über das Relais D und als Anker b vom +Pol der Batterie, die über das Relais 0 von den Leitern 1, 2, 3..., die den Stellen 1, 2, 3 ...entsprechen. Die Leiter 1, 2, 3 sind ferner noch über die Relais B und die Anker a an den +Pol angeschlossen. Die Relais B und C sind normalerweise offen, die Relais D geschlossen. Die Relais A erhalten also normalerweise ihren Strom über die Relais D und die Anker b.*

Soll nun eine Zahl gespeichert werden, so muß zuvor die in der betreffenden Zelle noch gespeicherte Zahl gelöscht werden. Das geschieht durch vorübergehendes Öffnen des Relais D, wodurch sämtliche Relais A stromlos werden. Jetzt wird Relais C geschlossen, die Relais , , somit an die Leiter 1, 2, 3 angeschlossen, über die die Zahl in das Speicherwerk gegeben wird. Die Relais werden geschlossen, falls die entsprechenden Leiter 1, 2, 3 Strom führen. Der Anker b wird angezogen und die geschlossenen Relais bleiben unter Strom, auch wenn das Relais C wieder geöffnet wird, während die nicht stromführenden Relais A den Anker b nicht anziehen und somit offen bleiben. Die Zahl ist gespeichert.

Soll die Zahl abgelesen werden, so wird Relais B geschlossen und die Leiter 1, 2, 3 über die Anker a an +Pol angeschlossen. Führen die Relais Strom, so ist die Verbindung geschlossen. Die Stromverteilung in den Leitern 1, 2, 3 entspricht also der gespeicherten Zahl. Die Relais B, C und D werden durch den Wähler gesteuert.

Das Wählwerk selektiert das Wort im Speicher, wo die 22 Bits abgelegt werden sollen.

1.2.5 Wählwerk der Maschine Z3

Die Auswahl einer Speicherzelle bzw. eines Wortes im Speicher erfolgt durch das Wählwerk. Die Adresse einer Speicherzelle wird durch eine 6-Bit Kodierung auf dem Lochstreifen angegeben. Die 6-Bit Kodierung steht rechts der Befehle Pr oder Ps. Abbildung 8 zeigt schematisch die Funktion des Wählwerkes.

Das Wählwerk ist ein Logikgatter, welches aus einer 6 Bit Kodierung (Adresse der Speicherzelle in binärer Darstellung) am Eingang Strom auf eine der 64 Leitungen am Ausgang zum Speicher legt. Die stromführende Leitung schaltet die 22 Speicherstellen (Bits) in einem Wort des Speichers so, daß es transportiert werden kann. Da mit 2^6 Bits genau 64 Dezimalzahlen dargestellt werden können, kann mit dem Wählwerk eine 6-Bits Kodierung auf 64 verschiedene Leitungen geschaltet werden. Die beiden in der Maschine Z3 vorgesehenen Befehle für den Speicher sind Pr z und Ps z. Der Buchstabe z gibt die Speicherzelle an. Der Befehl Pr liest den Inhalt des Wortes z in das Register R1 oder R2. Der Befehl

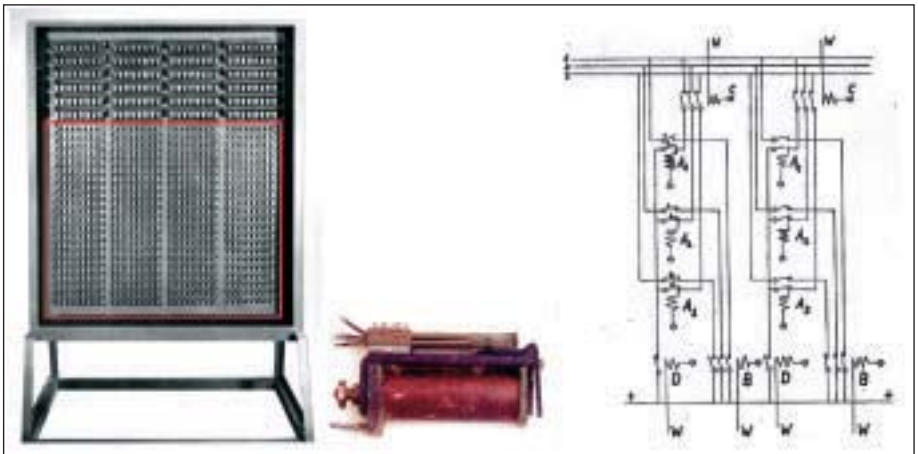


***Abbildung 7**: Links: Einer der beiden Speicherblöcke der Maschine Z3. Die umrandeten Relais (unteres Drittel) dienen zur Speicherung von 32 Worten a 22 Bits. Oberhalb des Speichers ist das Wählwerk angeordnet. In der Mitte ist ein Relais für ein Bit im Speicher der Z3 zu sehen. Ein Wort ist eine Spalte mit 22 Relais, die je ein Bit repräsentieren. Das oberste Bit ist das Vorzeichen. Das Wählwerk ist oberhalb des Speichers angeordnet und dient zur Ermittlung der Speicheradresse. Rechts: Eine Schaltung aus /ZUSE36/ für die Schaltung eines Speichers mit 3 Worten zu je drei Bits.*

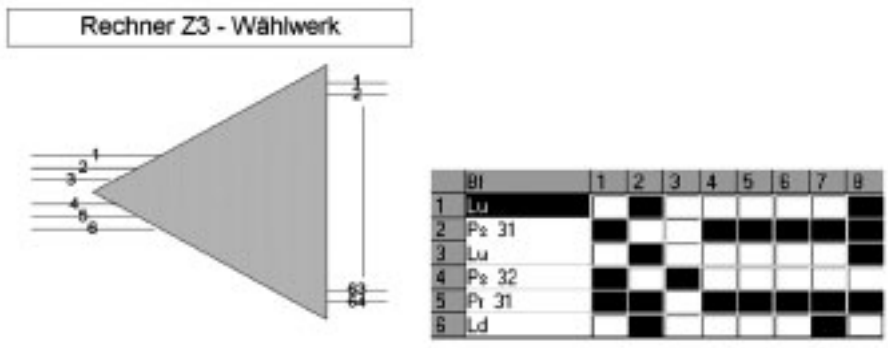


Abbildung 8: *Links: Symbolische Darstellung des Wählwerkes der Z3. Das Wählwerk ist ein Logikgatter, welches aus einer 6 Bit-Kodierung auf dem Lochstreifen (Adresse der Speicherzelle in binärer Darstellung) Strom auf eine der 64 Leitungen zu den 64 Worten im Speicher schaltet. Die stromführende Leitung steuert das betreffende Wort im Speicher an, indem es die betreffenden 22 Leitungen (Datenbus) schaltet. Rechts: Sechs Befehle auf dem Lochstreifen. Die Befehle Ps und Pr enthalten in den Bits 3-8 die binär kodierte Adresse für das Speicherwort.*

Pr 5 bewirkt z.B., daß der Inhalt der Speicherzelle 5 in das Register R1 geladen wird. Falls dieses besetzt ist, wird der Inhalt der Speicherzelle in das Register R2 geladen. Der Befehl Ps z transportiert den Inhalt des Registers R1 in das Wort z des Speichers. Der Inhalt von Register R1 wird danach gelöscht.

1.2.6 Arithmetische Einheit - Rechenwerk

Das Herz einer jeden Rechenmaschine ist die arithmetische Einheit, auch als Rechenwerk bezeichnet. Obwohl es um 1936 schon sehr leistungsfähige mechanische Tischrechenmaschinen gab, konstruierte Konrad Zuse ein Rechenwerk mit neuen, bisher nicht gekannten, sehr leistungsfähigen Komponenten, die von der Logik her auch heute noch in modernen Computern zu finden sind. Wir unterscheiden verschiedene Typen von Rechenwerken, z.B. Dezimalrechenwerke, binäre Festkommarechenwerke, binäre Gleitkommarechenwerke, usw. Konrad Zuse realisierte ein binäres Gleitkommarechenwerk, bestehend aus zwei binär arbeitenden Festkommarechenwerken (je eines für den Exponenten und die Mantisse).

Ein binäres Gleitkommarechenwerk verarbeitet binäre Gleitkommazahlen. Die Dezimalzahlen werden bei der Eingabe über die Eingabeeinheit in die normalisierte halblogarithmische Darstellung im Binärsystem umgewandelt und in eine binäre Mantisse mit Vorzeichen und einen binären Exponenten aufgeteilt. Wie oben dargestellt wurde, nannte Konrad Zuse dies die binäre halblogarithmische Zahlendarstellung. Die arithmetische Einheit der Maschine Z3 besteht aus zwei Rechenwerken, die beide arithmetische Operationen auf binären Zahlen ausführen können. Dies sind zwei Festkommarechenwerke. Eines behandelt die Rechenoperationen für den Exponenten und das andere behandelt die Rechenoperationen bezüglich der Mantisse

Das nächste Bild illustriert die Funktion der arithmetischen Einheit mit einem Blockschaltbild.

Das Rechenwerk hat Zugriff zu den beiden Registern R1 und R2, d.h. die binären Gleitkommazahlen für die Rechenoperationen liegen in den Registern R1 und R2 als Operanden für eine Rechenoperation bereit. Daher werden die beiden Register R1 und R2 auch als Operandenregister bezeichnet. Wir unterscheiden dyadische (+, -, *, /) und monadische (SQRT(x)) Rechenoperationen. Beide Register R1 und R2 werden bei dyadischen Operationen benötigt, während das Register R1 nur monadischen Rechenoperationen verwendet wird. Das Ergebnis einer Rechenoperation befindet sich immer in Register R1. Nach einer Rechenoperation wird Register R2 auf Null gesetzt (Nicht die Zahl Null, sondern alle 22 Bits auf 0).

Konrad Zuse führte alle Rechenoperationen auf wiederholte Additionen bzw. Subtraktionen (2-Komplement) zurück. Dazu verfügt das Rechenwerk über zwei leistungsfähige binäre Addierer, und zwar je einen für den Exponenten und die Mantisse der binären Gleitkommazahl. Der Aufbau des Gleitkommarechenwerkes ist allerdings weit komplizierter, als dies oben dargestellt wird. Die beiden Register R1 und R2 sind in Wirklichkeit nochmals aufgeteilt in Register für den Exponenten und die Mantisse. Auch existiert

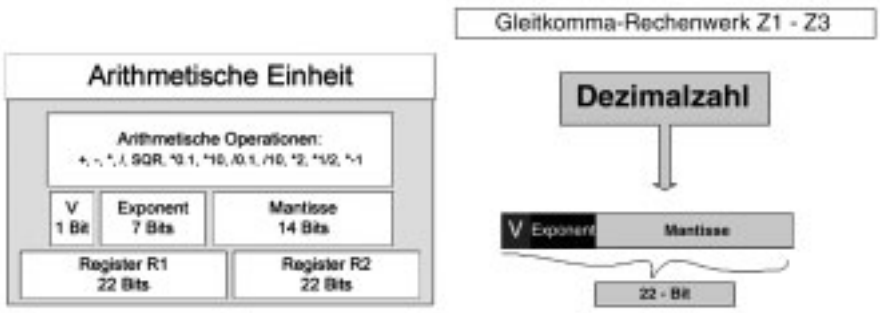


Abbildung 9: *Links: Die arithmetische Einheit der Z3 besteht aus ca. 600 Relais und Z3 arbeitet mit binären Gleitkommazahlen (halblogarithmische Zahlendarstellung). Die möglichen arithmetischen Operationen sind: +, -, *, /, SQRT(x). Eine Dezimalzahl, die über die Eingabeeinheit eingegeben wird, wird in der halblogarithmischen Form in einem Wort, welches wir auch Register nennen, gespeichert. Schrittschalter steuern den Ablauf der Rechenoperationen und führen sämtliche Rechenoperationen auf eine Folge von Additionen oder Subtraktionen zurück. Das Rechenwerk wird von der Steuereinheit gesteuert. Rechts: Das Gleitkommarechenwerk verarbeitet binäre Gleitkommazahlen mit 22 Bits. Mit V ist das Vorzeichenbit bezeichnet, der Exponent umfaßt 7 Bits und die Mantisse 14 Bits. Die Genauigkeit bei der Rechnung lag damit bei 14 / 3,5 = 4 Dezimalziffern.*

noch ein drittes Register, in welches das Ergebnis einer arithmetischen Operation geschrieben wird, bevor es in Register R1 zurückgeschrieben wird. Da z.B. die Multiplikation in eine Sequenz von Additionen aufgelöst wird, sind weitere Register notwendig, die als Zwischenspeicher arbeiten.

1.2.7 Das Problem der Addition- Der einschrittige Übertrag

Wir betrachten nun die Behandlung des binären Übertrages bei der Addition. Wir kennen dieses Problem bei der Addition von Dezimalzahlen. Der Übertrag in Form von zehn verschiedenen Ziffern muß von rechts nach links durchgereicht werden. Bei Binärzahlen ist die Situation etwas einfacher, da nur die binäre 1 von rechts nach links durchgereicht werden muß. Ein wichtiger Aspekt beim Entwurf des Addierers der Maschine Z3 war, daß er in der Lage ist, Additionen und Subtraktionen unter der Nutzung der Technik des *carry look-ahead,* auch einschrittiger Übertrag bezeichnet, durchzuführen. Konrad Zuse suchte nach einer Möglichkeit, das sequentielle Durchreichen eines binären Übertrages zu vermeiden und ersann eine Relaisschaltung, die den Übertrag für alle Binärstellen auf einen Schlag, d.h. mit einem Takt, berechnet.

Konrad Zuses einschrittiger Übertrag der Übertragungsbits (*carry-look-ahead*) bei der Addition zweier Binärzahlen ist unabhängig von der Länge der Binärzahlen. Damit hatte Konrad Zuse eine Konstruktion der Addition entwickelt, die es ihm erlaubte, beliebig lange Bitketten (Binärzahlen) zu addieren, ohne daß sich die Additionszeit verlängert. Abbildung 10 illustriert die Schaltung des einschrittigen Übertrages.

Die in Abbildung 10 dargestellte Schaltung arbeitet wie folgt: Die beiden zu addierenden Binärzahlen sind ba und bb. Wir bezeichnen mit ba0 - ba6 die Bits 0-6 der Binärzahl ba. Ebenso verfahren wir mit der Binärzahl bb. Zuerst werden die Bitpositionen bai und bbi durch die logische Operation bci = XOR(bai, bbi), die wir oben schon kennengelernt haben, logisch verknüpft. Danach werden bai und bbi durch die logische Operation bcai = AND (bai, bbi) verknüpft. Ist AND (bai,bbi) = 1, dann wird die Leitung, wo sich die Schalter bci befinden auf 1 geschaltet, d.h. stromführend. Die Schalter bci werden durch bci = XOR(bai,bci) geschaltet. Ist bci = 1, dann ist der Schalter geschlossen, bei 0 ist dieser offen. Je nachdem, ob die Schalter bci geschlossen oder offen sind und ob AND (bai,bbi) = 1 ist oder nicht, ist der eine Eingang bdi von bei = XOR(bdi,bci) auf 1 oder 0 geschaltet. Ist bdi = 1, dann ist ein Übertrag gesetzt, ist bdi=0, dann ist kein Übertrag gesetzt. Das Ergebnis einer Addition wird jetzt durch: be_i = XOR(bci,bdi) ermittelt

Die von Konrad Zuse erfundene Schaltung des binären Übertrages ist eine seiner faszinierendsten Ideen. In der Literatur ist bis heute nicht bekannt /LIEB98/, /LIEB01/, daß die ... *Zusesche Relaisschaltung ebenso wie die in der englischsprachigen Literatur unter „manchester carry chains" firmierenden MOS-Schaltungen in abstrahierender, logischer Symbolik, so wird die im Grunde gleiche Struktur deutlich.*

Wir illustrieren den einschrittigen Übertrag mit der Addition der Zahlen 107 und 119 in binärer Darstellung (siehe Abbildung 11).

Die beiden zu addierenden Binärzahlen ba und bb bestehen aus je 7 Bits. Mit ba_i und bb_i bezeichnen wir eine bestimmte Bitposition, z.B. mit ba_5 die Bitposition 5. In der ersten Binärzahl ba ist die Zahl 107 dargestellt, mit der zweiten Binärzahl ist es die Zahl 119. Das Ergebnis der Addition der beiden Binärzahlen ist 226 und steht in be ganz unten. Eine binäre 1 wird weiß, eine binäre 0 schwarz dargestellt. Wir erläutern nun Schritt für Schritt wie die Addition vorgenommen wird. Es sind 3. Schritte notwendig, um das Ergebnis zu erhalten, wobei die drei Schritte identisch zu drei Takten sind.

1. Takt: Für jede Bitposition berechnen wir bc_i = ba_i XOR bb_i. Mit dem tiefgestellten i sind die Bitpositionen, hier also 0 bis 7 gemeint. Das Ergebnis der logischen Operation XOR sehen wir in der Reihe bc = XOR (ba, bb). Gleichzeitig berechnen wir für jede Bitposition bca_i = AND (ba_i, bb_i). Das Ergebnis der logischen Operation steht in bca.

2. Takt: Der Wert von bci (0 oder 1) bestimmt, ob die Schalter zu denen die getrichelten Linien führen, geschlossen oder offen sind. Der Schalter wird geschlossen, wenn der Wert von bci = 1 ist. Wir sehen z.B., daß für die Bits 0 und 1 bc_0 = XOR (ba_0, bb_0) = 0 und bc_1 = XOR (ba_1, bb_1) = 0 jeweils 0 sind. Die Schalter bc_0 und bc_1 bleiben also

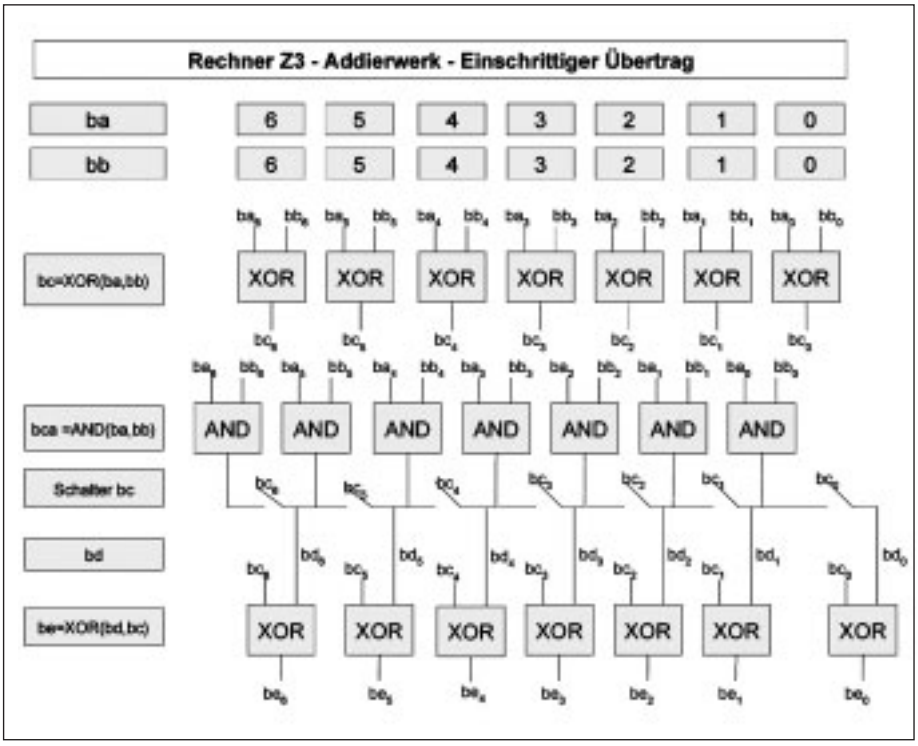


Abbildung 10: *Logisches Schaltbild des einschrittigen Übertrages zweier Binärzahlen ba und bb.*

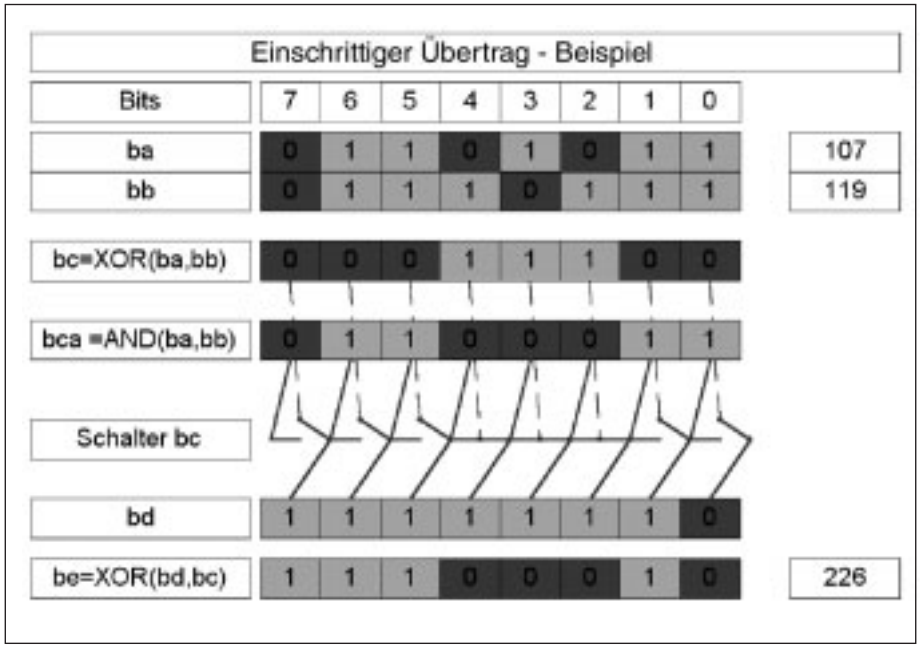


Abbildung 11: *Demonstration des einschrittigen Übertrages mit den Zahlen 107 und 119.*

offen. Ebenso verfahren wir mit bci = (XOR(bai, bbi) = 1. Ist bci = 1, wie z.B. bei bc_4, dann ist der entsprechende Schalter geschlossen.

Das Ergebnis von bca_i = AND (ba_i,bb_i) wird jetzt verwendet, um den Übertrag zu setzen. Ist bca_i = 1, dann werden die Leitung und bd_i auf 1 gesetzt. Durch die gesetzten Schalter in der Leitung werden auch andere bd_i Positionen auf 1 gesetzt, auch dann, wenn bca_i auf 0 steht.

3. Takt: Das Ergebnis der Addition zweier Binärzahlen ergibt sich dann aus be_i = XOR (bd_i, bc_i).

Wir sehen, daß Konrad Zuse den Übertrag bei der Addition von zwei Binärzahlen mit *einem Schlag*, d.h. parallel berechnet. Das faszinierende an dieser Schaltung ist, daß die Berechnung des Übertrages *invariant* (unabhängig) gegenüber der Länge (Bitanzahl) der Binärzahl ist. Wir wissen, daß die obigen Schaltungen nicht trivial sind. Aus unserer Sicht ist die Schaltung des einschrittigen Übertrages wichtig, da damit die Eleganz der Rechenwerke von Konrad Zuses Maschinen steht bzw. fällt.

Die Zurückführung der arithmetischen Operationen auf wiederholte Additionen bzw. Subtraktionen wird durch Mikrosequenzer, die mit Schrittschaltern realisiert sind, gesteuert. Sie sind im unteren Teil der arithmetischen Einheit angebracht. Die Bauteile werden als Schrittschalter bezeichnet und wurden früher in Telefonvermittlungszentralen als Endwähler bzw. Anrufsucher eingesetzt.

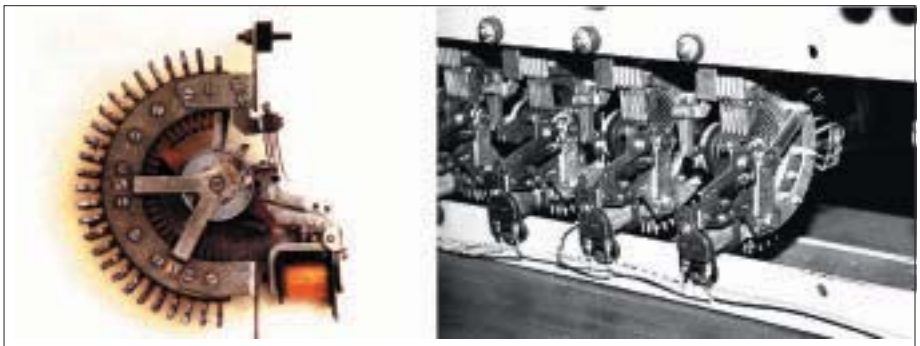


Abbildung 12: *Links: Ein Schrittschalter besteht aus einer Wicklung, einem Anker und einem Drehwerk, welches je nach Position mit Kontakten verbunden ist. Der Anker wird kurzeitig durch einen Stromstoß angezogen und bewegt das Drehwerk um einen Schritt weiter. Rechts: Die Schrittschalter in der Z3 zur Steuerung sequentieller Operationen.*

Konrad Zuse steuerte mit den Schrittschaltern die Folge der Additionen bzw. Subtraktionen bei den arithmetischen Operationen und die Konvertierung von binären Gleitkommazahlen in Dezimalzahlen und umgekehrt. Die Schrittschalter selbst wurden durch die Steuereinheit gesteuert. Die Schrittschalter wurden auch dazu verwendet, um die Konvertierung der Dezimalzahlen in die binäre halblogarithmische Darstellung und umgekehrt vorzunehmen.

1.2.8 Die Ein- und Ausgabeeinheiten der Maschine Z3

Wir betrachten nun die Ein- und Ausgabeeinheiten der Maschine Z3 (siehe Abbildung 13). Da Konrad Zuse seine Maschinen in der Praxis z.B. Ingenieurbüros einsetzen wollte, muß die Eingabe und Ausgabe der Zahlen in dem für Menschen gewohnten Dezimalsystem erfolgen.

Eine Übersetzung der Zahlen in der binären halblogarithmischen Darstellung in das Dezimalsystem ist bei der Resultatanzeige notwendig. Umgekehrt ist es notwendig, die Dezimalzahlen bei der Eingabe in die binäre halblogarithmische Zahlendarstellung zu übersetzen.

1.2.9 Arithmetische Ausnahmebehandlung

Konrad Zuses Vision war, daß seine Maschinen lange Rechnungen per Rechenplan vollautomatisch durchführen sollen. Dazu muß sichergestellt sein, daß die Maschine mathematisch korrekt rechnet, d.h. bei unzulässigen Rechenoperationen muß die Maschine stoppen und den Grund dafür anzeigen. Konrad Zuse konstruierte und implementierte dazu eine arithmetische Ausnahmebehandlung.

Stellt die Maschine Z3 eine unzulässige Rechenoperation fest, dann wird dies mit besonderen Lampen angezeigt. Links neben den vier Reihen von Lampen sind unter der Lampenanzeige Alarm sechs weitere Lampen angeordnet, die unzulässige Rechenoperatio-

Abbildung 13: *Links: Die Ein- und Ausgabeeinheit der Z3 (Nachbau). Vorne ist die Eingabetastatur für Dezimalzahlen zu sehen. Die Maschine Z3 kann auch als eine normale Tischrechenmaschine verwendet werden bzw. ein laufendes Programm kann durch Zwischenrechnungen im Dialog unterbrochen werden. Oben ist die Ausgabeeinheit zu sehen, die mit Lampen die Ergebnisse einer Rechnung als Dezimalzahl mit Mantisse und Exponent anzeigt. Rechts: Die Anzeige des Ergebnisses in dezimalen Gleitkommazahlen. Die Reihe der Lampen unten zeigt den Exponenten an und die vier parallelen Lampenreihen die Mantisse. Die ovalen Anzeigen auf der linken Seiten dienen zur Anzeige von arithmetischen Ausnahmesituationen.*

nen anzeigen. Auch die Zahl 0 ist ein Sonderwert, der aber nicht zum Abbruch einer Rechnung führt.

Das Ergebnis ist undefiniert:

- Unendlich plus (oder minus) Unendlich
- Null mal Unendlich
- Unendlich durch Unendlich
- Null durch Null

Diese arithmetischen Operationen sind undefiniert. Die Maschine Z3 stoppt die Ausführung des Rechenplans und die entsprechende Lampe leuchtet an der Ausgabeeinheit auf.

Das Resultat wird in folgenden Fällen unendlich:

- Bei Addition und Multiplikation, wenn einer der beiden Operanden Unendlich ist. Der Fall Null mal Unendlich wird gesondert gemeldet.
- Bei Division, wenn der Dividend Unendlich oder der Divisor gleich Null ist.
- Beim Quadratwurzelziehen, wenn der Radikand unendlich ist.

Das Resultat wird Null:

- Bei Addition, wenn beide Summanden gleich Null sind.
- Bei Multiplikation, wenn einer der beiden Faktoren gleich Null ist. Null mal Unendlich wird gesondert gemeldet.
- Bei Division, wenn entweder der Dividend gleich Null ist oder der Divisor Unendlich ist.
- Beim Wurzelziehen, wenn der Radikand gleich Null ist.

Kurz Zusammengefaßt:

- Überlaufen des Zahlenbereiches nach unten ergibt 0.
- Überlaufen des Zahlenbereiches nach unten ergibt ¥.
- Für das Rechnen mit 0 gilt: 0 + x = x, 0 * x = 0, x / 0 = ¥, 0 / 0 = ?
- Für das Rechnen mit ¥ gilt: ¥ + x = ¥, ¥ * x = ¥, 1 / ¥ = 0.

Alle Operationen, in denen ein ? ist, ergeben ?

Der Rechner Z3 verfügt damit über eine Behandlung von arithmetischen Sonderfällen, die nicht einmal in den ersten Mikroprozessoren der 80er Jahre implementiert waren.

1.3.10 Die vollständige Maschine Z3

Wir haben die einzelnen Konzepte bzw. Komponenten der historischen Maschine Z3 kennengelernt und deren Funktion hoffentlich verstehen gelernt. Es sind u.a. das verwendete binäre halblogarithmische Zahlensystem, die Steuereinheit, die Rechenplansteuerung, der Speicher mit dem Wählwerk, die arithmetische Einheit, der einschrittige Übertrag, die Ein- und Ausgabeeinheiten, und der Impulsgeber. Alles das fügt sich zu der funktionierenden Maschine Z3 zusammen, die damit wesentliche Eigenschaften des modernen Computers enthält:

- Freie Programmierbarkeit der Maschine mit einem 8-Kanal-Lochstreifen (35 mm Kino-Normalfilm). Auf dem Lochstreifen können je 64 verschiedene Speicherzellen für Lese- und Schreiboperationen adressiert werden und fünf arithmetische Operationen und zwei Ein- / Ausgabeoperationen eingestanzt werden. Der Lochstreifen kann damit neun plus 128 verschiedene Befehle enthalten.
- Klare Trennung von Kontrolleinheit, Speichereinheit, arithmetischer Einheit, und den Ein- und Ausgabeeinheiten. Verbindung dieser Einheiten mit einem parallel arbeitenden Datenbus von 22 Bits. Dies bedeutet eine parallel arbeitende Maschine mit 22 Bits Datenübertragung.
- Implementierung von Mikrosequenzen (Schrittschalter) in der Steuereinheit, d.h. Rückführung der arithmetischen Operationen auf Sequenzen elementarer Operationen, wie Addition und Subtraktion.
- Binäre arithmetische Einheit im Gleitkommasystem mit den vier Grundrechenarten und den zusätzlichen Operationen: SQR(x). Im Jahr 1985 standardisierte IEEE im Standard 741 die Darstellung von Gleitkommazahlen. Auffallend ist die fast identische Darstellung von Gleitkommazahlen im IEEE-Report und in den Maschinen Z1 (1936-1938) und Z3 (1941).
- Der einschrittige Übertrag bei der Addition- und Subtraktion (Addition in drei Takten).
- Umwandlung der binären Gleitkommazahlen in Dezimalzahlen und umgekehrt in den Ein- und Ausgabeeinheiten.
- Binäre Schaltungslogik (Schaltalgebra) in der gesamten Maschine (UND, ODER, NEGATION).

Es fügt sich nun alles zusammen und Konrad Zuse hatte damit 1941 eine funktionierende vollautomatische Rechenmaschine, die frei programmierbar ist und auf dem binären Gleutkommazahlensystem arbeitet, konstruiert. Es gab um diese Zeit keine andere vergleichbare Maschine dieser Art welt-

weit. Die Frage ist: Was fehlt der Maschine Z3 zum modernen Computer oder was ist der Unterschied eines Computers zu einer programmgesteuerten Rechenmaschine?

Die Maschine Z3 kann nur starre Rechenpläne verarbeiten, d.h. die einmal im Rechenplan festgelegten Befehle sind auf einen 35mm Kino-Normalfilm gespeichert und können nicht verändert werden. Keine andere Maschine zur damaliger Zeit konnte die Befehle im Speicher der Maschine selbst modifizieren. Der Ablauf des Rechenplans der Maschine Z3 kann nicht verändert werden, nur die Eingangs- und Zwischenwerte im Speicher können geändert werden. Auch verfügt die Maschine Z3 nicht über den bedingten Sprung, d.h. eine Bedingung bzw. einem Befehl, der es abhängig von errechneten Zwischenwerten ermöglicht, den Ablauf des Rechenplanes auf dem Lochstreifen zu ändern. Das Fehlen des bedingten Sprunges in der Maschine Z3 ist überraschend, denn die Implementierung wäre sehr einfach gewesen. Bei der Maschine Z4 wurde der bedingte Sprung 1949 auf Wunsch der ETH-Zürich (Eidgenössische Technische Hochschule) realisiert: Abhängig vom Inhalt des Register R1 werden bei der Z4 die nachfolgenden Befehle auf dem Lochstreifen bis zu einem bestimmten Nachfolgebefehl überlesen.

Aus unserer Sicht sah Konrad Zuse bei der Maschine Z3 im Jahr 1941 keine Notwendigkeit, den bedingten Befehl zu realisieren. Die von ihm untersuchten wissenschaftlichen Aufgaben erforderten keinen bedingten Befehl. Konrad Zuse wurde in der Vergangenheit immer wieder gefragt, warum er den bedingten Sprung in der Z3 nicht realisierte. Seine Antworten waren z.B:

- *Ich wollte den Draht für den bedingten Sprung nicht legen.*
- *In meinen Tagebuchnotizen habe ich schon 1938 lebendige Rechenpläne eingeführt, die den Ablauf des Rechenplans abhängig von den Zwischenwerten ändern können.*
- *Im Plankalkül(1942-1945) ist der bedingte Sprung enthalten.*
- *Während dieses Krieges und der Bombenangriffe blieb nicht viel Zeit für innovative Ideen und schon gar nicht für die Konstruktion von noch komplexeren Maschinen.*

Es ist korrekt, daß Konrad Zuse 1938 /ZUSE38/ das Rechnen mit Umständen und Bedingungen beschrieb. Dazu führte er die *lebenden Rechenpläne* ein, die er 1943 /ZUE43/ in *freie Rechenpläne* umdefinierte. Es ist keine Frage, daß Konrad Zuse um 1943 den in der Maschine *verlöteten* bzw. *fest verdrahteten* bedingten Sprung herauslösen und auch als eigenständigen Befehl zur Verfügung stellen wollte. Aus unserer Sicht wollte er die schon sehr komplexe Maschine Z3 durch die Möglichkeit des bedingten Sprungs nicht noch komplexer gestalten. Daß er den bedingten Sprung kannte, steht zweifelsfrei fest, da er diesen z.B. in der arithmetischen Einheit fest verdrahtet implementiert hatte. Ein Gleitkommarechenwerk ist ohne den bedingten Sprung nicht zu realisieren. Der Plankalkül, dessen Endfassung er 1945/46 in Hinterstein schrieb, enthält den bedingten Sprung. Die Arbeiten für den Plankalkül begannen nach seinen eigenen Aussagen gegenüber dem Autor um 1942.

Die Speicherung des Rechenplanes im Speicher der Maschine Z3 mit einer möglichen Adreßumrechnung für die Zwischenwerte wäre aufwendig gewesen, da jeder Befehl wertvollen Speicherplatz, der zu dieser Zeit teuer und knapp war, beansprucht hätte. Die Idee der freien Programmierbarkeit der Maschine wäre durch die beschränkte Befehlszahl eingeschränkt worden.

2 Plankalkül - Die Symbiose zwischen Hard- und Software

Mit seinem in den Jahren 1942-1945 entwickelten Programmiersystem, dem Plankalkül, wollte Konrad Zuse schwierige Aufgaben der Ingenieure, wie z.B. aus dem Bauwesen, in Programme fassen. Sein Plankalkül enthielt die weit über das pure Zahlenrechnen hinausgehenden *Regeln des logischen Schließens* der mathematischen Logik. Im Plankalkül finden wir u.a. folgende Sprachkonstrukte: Zuweisungszeichen, mächtige hierarchische Datenstrukturen, Datentypen wie Gleitkommazahlen, Festkommazahlen, komplexe Zahlen, Unterprogrammtechnik mit Schnittstellendefinition, Bedingte Anfragen, sieben verschiedene Schleifenarten (u.a. die WHILE-Schleife), Listenverarbeitung, Relationen, Prädikatenkalkül, arithmetische Ausnahmebehandlungen und sechzig Seiten Schachprogramme. Für Konrad Zuse war klar, daß künftige Rechner Aufgabenstellungen aus der Kombinatorik (In Zuses Worten: alle *rechenbaren* Probleme) lösen sollten. Er bezeichnete diese Maschinen als logistische Rechenmaschinen im Gegensatz zu den algebraischen Maschinen wie z.B. die Z3.

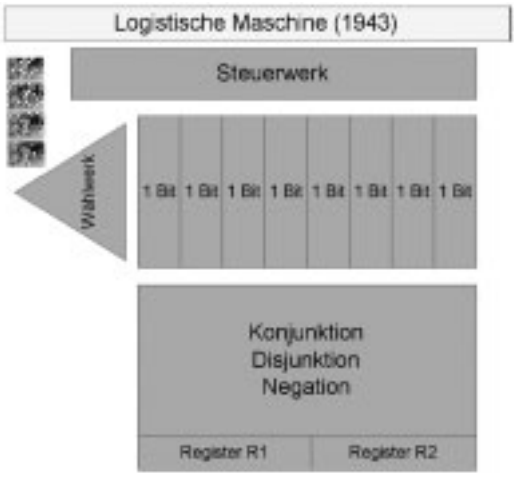


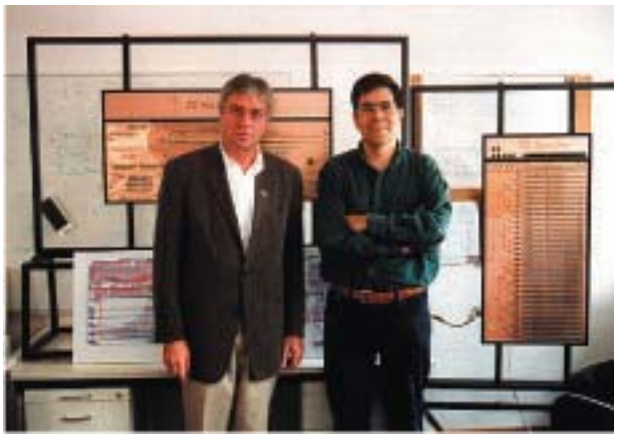
Abbildung 14: *Logistische Maschine (1943). Die Architektur der logistischen Maschine ist denkbar einfach aufgebaut. Sie besteht aus einem Speicher mit Zellen mit jeweils einem Bit und einer logischen Einheit mit den Operationen UND, ODER und NICHT. Konrad Zuse hat diese Maschine nie gebaut, lediglich eine Versuchsschaltung wurde um 1944 erstellt.*

Es ist bemerkungswert, daß Konrad Zuse zwei Typen von Rechenmaschinen entwickelte. Der eine Typ von Rechenmaschine war die algebraische Rechenmaschine, wie z.B. die Maschinen Z1, Z3 oder Z4, die nur zum Zahlenrechnen geeignet waren. Die logistische Maschine sollte im Gegensatz zu den algebraischen Maschinen logische Operationen durchführen können. Die Architektur der logistischen Rechenmaschine ist denkbar einfach. Es gibt einen Speicher mit sehr vielen 1-Bit Zellen. Dazu kommt ein Rechenwerk, welches nur die logischen Operationen UND, ODER und NICHT ausführen kann. Eine Programmsteuerung und das Steuerwerk komplettieren die Maschine. Die Ähnlichkeit dieser Ma-

schine mit der Turing-Maschine von 1936 ist auffallend. Konrad Zuse kannte allerdings 1943 nicht die Arbeiten von Alan Turing in UK. Heutzutage sind beide Maschinentypen vereint, wie z.B. im PENTIUM-Prozessor. Konrad Zuse wollte die Aufgaben der komplexen Hardware in die Software (Plankalkül) verlagern.

3 Der Nachbau der Maschine Z3 vom 11. Mai 2001

Der Nachbau der Rechenmaschine Z3, der auf dem Workshop am 11. Mai vorgestellt wurde, ist ein Projekt unter der Leitung von Dr. Horst Zuse (TU) und Prof. Raul Rojas (FU). Die elektronischen Schaltungen wurden von Dr. Frank Darius (FU) und Dipl.-Ing. Georg Heyne (Fritz-Haber-Institut Berlin) in mühevoller Arbeit entworfen und implementiert. Auch Schüler haben ihren Beitrag für dieses Vorhaben geliefert: Der Rahmen für die Maschine wurde von der 1. Berufsschule für Sonderpädagogik (Berlin-Pankow), die Konsole vom Friedrich-Schiller-Gymnasium (Bautzen) und der Lochstreifenleser von der Konrad-Zuse-Schule (Hünfeld) unter Leitung des Lehrers Uwe Trautrims gebaut.



Dr. Horst Zuse und Professor Dr. Raul Rojas vor dem Nachbau der Z3.

Dr. Horst Zuse, Professor Dr. Raul Rojas und Bundespräsident Rau bei der Besichtigung des Nachbau der Maschine Z3 im Wissenschaftszentrum in Bad Godesberg am 13. Mai 2001.

Am 13. Mai 2001 beging das Wissenschaftszentrum Bonn - Sitz der GI-Geschäftsstelle - sein 25-jähriges Gründungsjubiläum mit einer großen Schar prominenter Gäste, darunter NRW-Ministerpräsident Wolfgang Clement, Staatssekretär Dr. Uwe Thomas (Bundesministerium für Bildung und Forschung), die Oberbürgermeisterin der Stadt Bonn, Bärbel Diekmann, und schließlich Bundespräsident Dr. h.c. Johannes Rau. Um den Besucherinnen und Besuchern einen Einblick in die vielfältigen Einsatzmöglichkeiten der Informatik zu geben, hatte die GI gemeinsam mit der Konrad-Zuse-Gesellschaft ein interessantes Programm zusammengestellt. Den Blickfang bildete neben verschiedenen Fußball spielenden

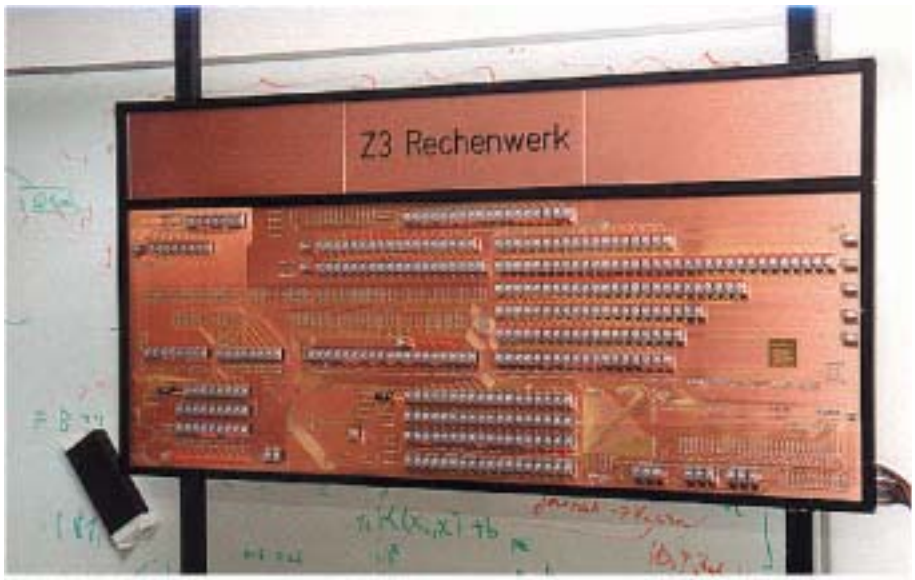


***Foto oben**: Nachbau der Z3 vom 11. Mai 2001, bei dem die Bits leuchten. Links das Rechenwerk der Z3 und rechts der Speicher der Z3 mit 32 Worten a 22 Bits. Der Speicher besteht aus 32 horizontal angeordneten Worten an 22 Bits. **Foto unten**: von den 32 Worten ist das Wählwerk angeordnet, welches auf Basis einer 5-Bit-Kodierung die Adresse 1-32 im Speicher ermittelt. Die Funktion des Wählwerkes kann man sich wie folgt vorstellen: Wir haben einen Bahnhof mit 32 Gleisen. Von links kommt eine ICE und soll auf eines der Gleise geleitet werden. Dies geschieht durch die Schaltung von Weichen, die entweder nach links oder rechts geschaltet werden können. Wollen wir ein Wort im Speicher ablegen, dann werden zur Schaltung Relais verwendet.*

und Marslandschaften erklimmenden Robotern die zum 60. Geburtstag des Computers nachgebaute Rechenmaschine Z3 von Konrad Zuse, deren Funktionsweise sein Sohn, Horst Zuse und Raul Rojas, den interessierten Besucherinnen und Besuchern erläuterte. Sogar Bundespräsident Johannes Rau begeisterte sich für den Nachbau. So konnten Raul Rojas und Horst Zuse den Bundespräsidenten am Stand von GI und Zuse-Gesellschaft begrüßen und ihn in die Geheimnisse des Computerbaus einweisen. Zum Abschied überreichte Horst Zuse Johannes Rau die Autobiographie seines Vaters „Der Computer - Mein Lebenswerk“.

Der Nachbau der Rechenmaschine Z3 ist mit kleinen Relais realisiert worden. Die Architektur der Maschine wurde respektiert, aber so umgesetzt, dass der Fluß von Daten durch kleine Leuchtdioden angezeigt wird. Damit ist der Nachbau historisch korrekt und gleichzeitig pädagogisch geeignet, um Schülern und Studierenden die Geschichte dieser Rechenmaschinen lebendig werden zu lassen. Die Maschine ist klein genug, um transportiert zu werden, und wird in der Zukunft bei verschiedenen Ausstellungen gezeigt. Eine Rekonstruktion im Ein-zu-Eins-Maßstab wurde in den sechziger Jahren von Konrad Zuse selbst gebaut und befindet sich heute im Deutschen Museum in München.

4 Abschließende Bemerkungen

Konrad Zuses Ideen über das Rechnen und die Anwendung seiner vollautomatischen Rechenmaschinen gehen weit über das pure Zahlenrechnen hinaus. Das Bauingenieurwesen nahm dabei eine besondere Stellung ein. Besonders erwähnenswert ist die von Konrad Zuse durchgeführte Symbiose von Hardware und Software schon im Jahr 1945 mit Hilfe der ersten höheren Programmiersprache, dem Plankalkül. Die Kombination der algebraischen Rechengeräte, wie die Maschinen Z3 und Z3 mit der logistischen Maschine, die zur Behandlung von logischen Operationen konzipiert war, als sinnvolle Abrundung des Ganzen das Manuskript des Plankalkül (1945) sind eine sehr elegante Symbiose von Hard- und Software, die in dieser Form um 1945 weltweit nicht vorhanden war.

Literatur

/ALEX97/ Alex, Jürgen: Wege und Irrwege des Konrad Zuse. Spektrum der Wissenschaft, Januar 1997.

/BAUE98/ Bauer, Friedrich, L.: Wer erfand den Neumann'schen Rechner? Informatik Spektrum, April 98, pp. 84ff.

/BEAU68/ Beauclair de, W.: Rechnen mit Maschinen. Vieweg Verlag, 1968. (Kommentar: Dieses Buch von 1968 ist eine exzellente und einzigartige Übersicht über die Historie der Rechnerentwicklung in Wort und Bild.)

/BURK46/ Burks, A.W.; Goldstine, H.H. Neumann, John von: Preliminary Discussion of the Logical Design of an Electronical Computing Instrument, 1946. In: Taub, A.H. (Editor), Collected Works of John von Neumann, Vol. 5, New York, Macmillan, 1963, pp. 34-79. Also in Randell 1973 /RAND73/, pp. 371ff.

/CERR83/ Ceruzzi,, Paul: Reckoners - The Prehistory of the Digital Computer from the Relays to the Stored Program Concept, 1935-1945. Greenwood Press, Westport, Conneticut, London, England, 1983.

/CZAU79/ Czauderna, K.-H.: Konrad Zuse, der Weg zu seinem Computer Z3, Oldenburg Verlag, München, 1979. (Kommentar: Dieses Buch ist eine frühe Würdigung der Arbeiten von Konrad Zuse. Es ist u.a. deswegen besonders wertvoll, da in dem Buch Augenzeugenberichte über die Entwicklung der Z3 enthalten sind.)

/DORS89/ Dorsch, Hadwig: Konrad Zuses Z1 – Berlin 1936. Museum für Verkehr und Technik, 1989.

/IEEE85/ IEEE: IEEE Standard for Binary Floating Point Arithmetic (ANSI / IEEE Std 754-1985), New York, August 1985.

/LIEB98/ Liebig, Hans: Brief an Horst Zuse zu den Themen: Einschrittiger Übertrag, Neumann'scher Rechner .

/LIEB00/ Liebig, Hans; Menge, M.: Zuse und der nicht-einschrittige Übertrag. Informatik Spektrum, Dezember 2000, pp. 398-401.

/NEUM45/ Neumann, John von: First Draft of a Report on the EDVAC. Moore School of Electrical Engineering, University of Pennsylvenia, June 30, 1945. In /RAND73/, pp. 355ff.

/RAND73/ Randell, B. (Editor): The Origins of Digital Computers. Springer Verlag, Heidelberg, New York, 1973.

/ROJA97/ Rojas, Raul: Konrad Zuse's Legacy: The Architecture of the Z1 and Z3. IEEE Annals of the History of Computing, Vol. 19, No. 2, 1997.

/ROJA98/ Rojas, Raul: How to make Zuse's Z3 a Universal Computer. IEEE Annals of Computing, Vol. 20, No. 3, July/Sept. 1998.

/ROJA98a/ Rojas Raul (Editor): Die Rechenmaschinen von Konrad Zuse, Springer Verlag, 1998.

/ROJA00/ Rojas, Raul: Konrad Zuse - Elektronisches Archiv. Internet: http://www.zib.de/zuse, 2000.

/SCHR41/ Schreyer, Helmut: Das Röhrenrelais und seine Schaltungstechnik. Dissertation an der TU-Berlin, 9. Oktober 1941, Berichter: Prof. Dr.-Ing. Stäblein, Mitberichter: Prof. Dr. H. Fassbender.
Kommentar: In dieser Dissertation werden elektronische Relaisschaltungen gezeigt, die Schreyer, ein Freund Konrad Zuses, entwickelt hat. Wir finden die Elektronenröhre als Relais, das Stromtor als Relais, Röhrenrelais als Nachbildung eines elektromagnetischen Relais mit Schließkontakt, Röhrenrelais mit Schließkontakt, Röhrenrelais mit mehreren Glimmlampen entsprechend einem elektromagnetischen Relais mit mehreren Kontakten und das Röhrenrelais als Selbsthaltekreis.

/SCHW89/ Schweier, U. und Saupe, D.: Funktions- und Konstruktionsprinzipien der programmgesteuerten Rechenmaschine Z1, Arbeitspapiere der GMD, 321, 1988.
Kommentar: Schweier und Saupe unterstützten Konrad Zuse beim Nachbau der Maschine Z1 von 1986-1989. Obige Arbeit (Diplomarbeit) beschreibt einige Konstruktionsprinzipien der Z1. Die Arbeit ist ein wertvolles historisches Dokument.

/SCHU98/ Schunke, Katja: Zur Rezeption des Plankalküls von Konrad Zuse - Die Einordnung in die Programmierprachenkonzepte zur Zeit seiner Veröffentlichung, Diplomarbeit an der TU-Berlin, Prof. Siefkes, 1998.

/SPEI50/ Speiser, Ambros: Das Programmgesteuerte Rechengerät Z4 an der Eidgenössischen Technischen Hochschule in Zürich. Separatabdruck der Beilage Technik aus der Neuen Züricher Zeitung, Nr. 1796 (50) vom Mittwoch, den 30. August 1950.

/SPEI98/ Speiser, Ambros: The Early Years of the Institute: Aquisition and Operation of the Z4, Planning of the ERMETH. Departement of Computer Science, ETH-Zürich, 1998.

/ZUSE36/ Zuse, Konrad: Verfahren zur selbsttätigen Durchführung von Rechnungen mit Hilfe von Rechenmaschinen. Patentanmeldung Z 23 139 / GMD Nr. 005/021 / Jahr 1936.

/ZUSE37a/ Zuse, Konrad: Einführung in die allgemeine Dyadik., 1937.

/ZUSE38/ Zuse, Konrad: Tagebuchnotiz vom 4.6.1938.
Kommentar: : Hier wird die Computerarchitektur in verschiedenen Phasen bis zur heute als John v. Neumann-Maschine bezeichneten Anordnung beschrieben.

/ZUSE39/ Zuse, Konrad: Rechenmaschine. Bericht von Konrad Zuse als Gefreiter, 1939.

/ZUSE43/ Zuse, Konrad: Rechenplangesteuerte Rechengeräte für technische und wissenschaftliche Anwendungen, 1943.
Kommentar: In diesem Artikel werden u.a. logistische Maschinen und

freie Rechenpläne eingeführt. Auch wird über das Rechnen mit Umständen und Bedingungen berichtet (Bedingter Sprung).

/ZUSE45/ Zuse, Konrad: Theorie der angewandten Logistik - 2. Buch. der Zuse Apparatebau Berlin, 1945.

/ZUSE47/ Zuse, Konrad: Über die Mechanisierung Schematisch - Kombinativer Aufgaben. Zuse Ingenieurbüro Hopferau, 1947.

/ZUSE48/ Zuse, Konrad: Über den Plankalkül als Mittel zur Formulierung schematisch-kombinativer Aufgaben. In: Archiv Mathematik, Band I (1948/49).

/ZUSE49d/ Zuse, Konrad: Rechenmaschine (Logarithmisch). Patentanmeldung, 1949.

/ZUSE67a/ Zuse, Konrad: Die Entwicklung der programmgesteuerten Rechenanlagen. Bad Hersfeld, ca. 1967.

/ZUSE69/ Zuse, Konrad: Rechnender Raum, Schriften zur Datenverarbeitung Band 1. Friedrich Vieweg & Sohn, Braunschweig (1969).

/ZUSE72/ Zuse, Konrad: Der Plankalkül. Gesellschaft für Mathematik und Datenverarbeitung. Nr. 63, BMBW - GMD - 63, 1972.

/ZUSE72a/ Zuse, Konrad: Zur Problematik der Rechenautomaten. Wissenschaftliche Buchgesellschaft Darmstadt, 1972.

/ZUSE86/ Zuse, Konrad: Der Computer - Mein Lebenswerk. 2. Auflage, Springer-Verlag, 1986.

/ZUSE93/ Zuse, Konrad: The Computer – My Life. Springer Publisher, 1993.

/ZUSE99b/ Zuse, Horst: Geschichte der Programmiersprachen. Technischer Bericht 1/99, TU-Berlin, 1999.

/ZUSE99c/ Zuse, Horst: Konrad Zuse - His Life and Work. Internet: http://www.epemag.com/zuse. September 1999.

/ZUSE00/ Zuse, Horst: Konrad Zuse Multimedia Show, Version 2.0. April 2000. Erhältlich unter der URL: http://home.t-online.de/home/horst.zuse.

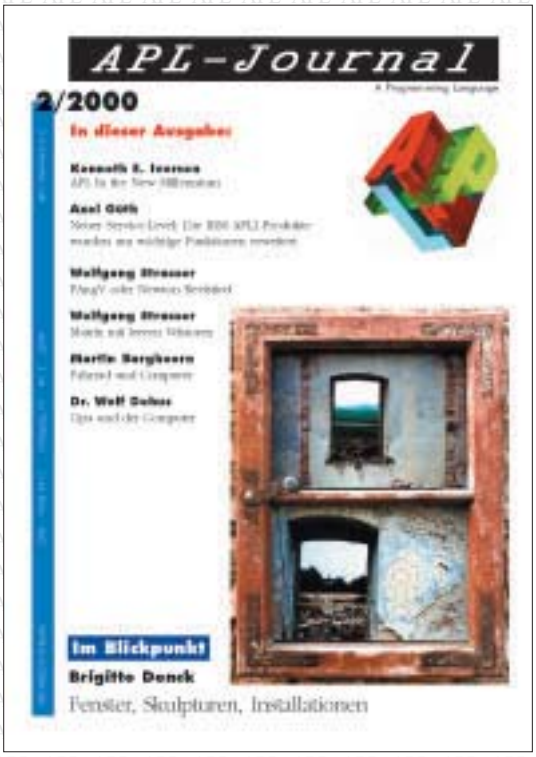
Autor

Dr.-Ing. Horst Zuse wurde 1945 geboren und studierte von 1967 bis 1973 Elektrotechnik an der Technischen Universität Berlin (TUB). Im Jahr 1985 promovierte er auf dem Gebiet der Softwarekomplexitätsmaße und es folgte 1987 ein einjähriger Forschungsaufenthalt am *IBM Thomas J. Watson Research Center in Yorktown Heights* / New York in den USA. 1991 publizierte er das Buch: *Horst Zuse: Software Complexity - Measures and Methods,* im DeGruyter Verlag und 1998 erschien das Buch: *Horst Zuse: A Framework of Software Measurement, DeGruyter Verlag.* Vom Februar bis Mai 1998 war er als Visiting Professor an der *University of Southwestern Louisiana* in Lafayette / USA tätig. Im Jahr 1998 verlieh der Fachbereichsrat Informatik der TU-Berlin ihm im Rahmen des Habilitationsverfahrens die Lehrbefähigung auf dem Gebiet der *Praktischen Informatik.* Im Jahr 2000 veröffentlichte er das Buch: *Alex; Flessner; Mons; Pauli; Zuse: Konrad Zuse - Der Vater des Computers.* Im Jahr 2000 wurde er Mitgründer der StartUp-Firma *PixFind Net Technologies.*

Kontakt

Dr. Horst Zuse, TU-Berlin, Franklinstraße 28/29, D-10587 Berlin, Tel. 030-881 59 88, Fax: 030-886 81 678, E-mail: horst.zuse@t-online.de, Internet: http:// www.zuse.org.

APL - Germany e.V.



Hier erhalten Sie das APL-Journal in Farbe per Download (PDF-Datei): www.rhombos.de\apljourn.htm

RHOMBOS-VERLAG ✆ ++49-(0)30-2616854 eMail: verlag@rhombos.de Internet: www.rhombos.de

Conrad Hoesle-Kienzlen

Vom Einzelplatz ins Internet

Die Fallstricke bei der Client/Server-Programmierung

Die bis heute üblichen APL-Anwendungen laufen im Normalfall für jeden Benutzer getrennt in einem eigenen Adressraum. Das heißt, für jeden Benutzer wird ein eigenes APL gestartet, in dem die Anwendung exklusiv für diesen Benutzer läuft. Alle Variablen, die den Stand der Anwendung zu einem beliebigen Zeitpunkt widerspiegeln, sind ständig vorhanden und ihr Zustand ändert sich nur auf Grund von Benutzer-Interaktionen. Das Gleiche gilt für den Statusindikator.
In einer Client/Server-Umgebung ist das fundamental anders. Hier können mehrere Benutzer gleichzeitig das selbe Programm in einem einzigen Adressraum verwenden. Es muss also sichergestellt sein, dass jeder Benutzer zu jedem Zeitpunkt seine individuelle Umgebung vorfindet und nicht zufällig die Umgebung eines anderen Benutzers verändert.
Dieser Beitrag beruht auf Erfahrungen, die bei der Umstellung von alten APL-Anwendungen auf Client/Server-Technik gemacht wurden. Die aufgetretenen Probleme werden geschildert und die dafür entwickelten Lösungen aufgezeigt. Diese sind im Softwarepaket Web4APL enthalten, das vom Autor erstellt wurde und über die Firma Dittrich & Partner vertrieben wird.

Einzelplatz- und Client-Server-Paradigma

Einzelplatz-Anwendungen folgen einem völlig anderen Paradigma als Client-Server Anwendungen. Um zu verstehen, was es bedeutet, eine Einzelplatz-Anwendung auf den Betrieb in einem Web-Server umzustellen, muss man sich das Flussdiagramm der beiden Umgebungen anschauen.

Bei der Einzelplatz-Anwendung wird das Hauptprogramm gestartet. Dieses baut eine Bildschirm-Maske auf, stellt sie auf dem Bildschirm des Benutzers dar und wartet auf die Benutzereingaben. Wenn der Benutzer seine Eingaben gemacht hat, werden diese vom Anwendungsprogramm analysiert, eine neue Bildschirm-Maske aufgebaut, diese dargestellt und wieder gewartet.

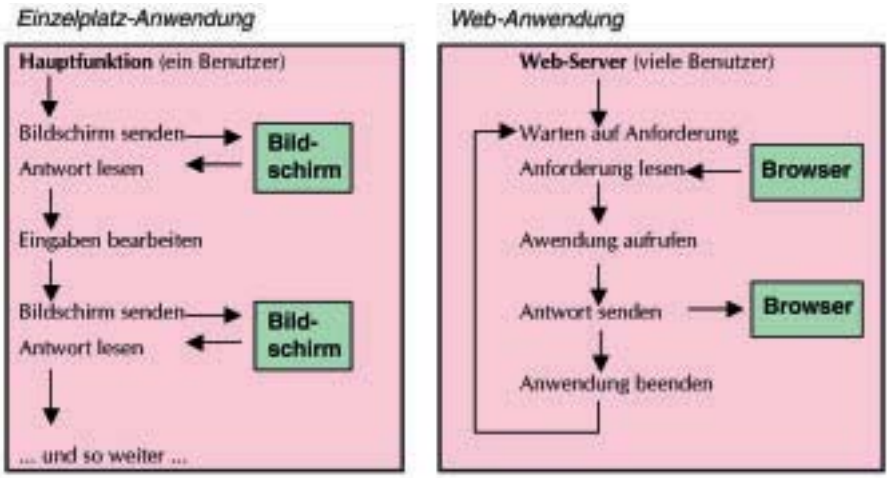


Abbildung 1: *Flußdiagramm von Einzelplatz- und Client-Server- Anwendungen*

Bei jedem Schritt hat das Anwendungsprogramm vollen Zugriff auf alle Daten, die bisher eingegeben wurden, und weiss, was als nächstes zu tun ist.

➪ Das Anwendungsprogramm hat die Kontrolle über den gesamten Ablauf.

Ein Web-Server benutzt ein statusloses Protokoll. Der Server erhält eine Anforderung, versucht, diese zu erfüllen und sendet die Antwort an den Client. Damit ist die Transaktion beendet und der Server wartet auf die nächste Anforderung. Diese kann von einem ganz anderen Benutzer kommen, der eine völlig andere Anforderung stellt. Der Server hat keine „Erinnerung" an vorherige Anforderungen und macht keinen Unterschied zwischen verschiedenen Benutzern. Für diejenigen, die sich noch daran erinnern können: Ein Web-Server arbeitet ähnlich wie CICS.

➪ Das Programm, das den Ablauf kontrolliert, ist der Server, nicht die Anwendung.

Die Fallstricke

Aus dem oben Gesagten ergeben sich diverse Problemkreise:

- Die Bildschirm-Masken müssen in HTML umgesetzt werden.
- Es muss sichergestellt sein, dass jeder Benutzer auch die Berechtigung besitzt, die Anwendung aufzurufen.
- Da ein Web-Server keine Erinnerung an vorhergehende Transaktionen hat, muss diese notwendige Erinnerung in das Anwendungsprogramm eingebaut werden.
- Wenn die gleiche Anwendung viele Benutzer bedienen soll, darf eine einzelne Transaktion nicht zu lang dauern, damit die anderen Benutzer nicht zu lange warten müssen.
- Benutzerbezogenes Dateihandling
- Besondere Vorkehrungen sind nötig, wenn auch APL-Zeichen eingegeben werden sollen.
- Was soll im Falle eines Programmfehlers passieren?

Wie sehen nun die Lösungen zu diesen Problemen aus?

Bildschirm-Masken und HTML

Für dieses Problem gibt es mehrere Lösungen.

Man kann das Programm *WebConverter* einsetzen, welches AP124/126-Masken in HTML umsetzt.

Diese Methode ist sehr einfach anzuwenden, dabei geht jedoch der Vorteil verloren, typische HTML-Elemente wie Drop-Down-Listen, Radiobuttons und Checkboxen einsetzen zu können.

Für Anwendungen, die längerfristig im Web eingesetzt werden sollen, sollte man sich die Mühe machen, die Masken in richtiges HTML umzuarbeiten und die Gestaltungsmöglichkeiten von HTML, StyleSheets und JavaScript einsetzen.

Wenn es um statische Masken geht, d.h. der Benutzer dort keine Eingaben machen kann oder soll, kann man eine HTML-Datei erstellen (z.B. mit einem HTML-Editor wie Dream-Weaver). Diese Datei wird dann vom Web-Server an den Browser geschickt. In der HTML-Seite können z.B. Links enthalten sein, auf die der Benutzer klickt, um verschiedene Optionen der Anwendung aufzurufen.

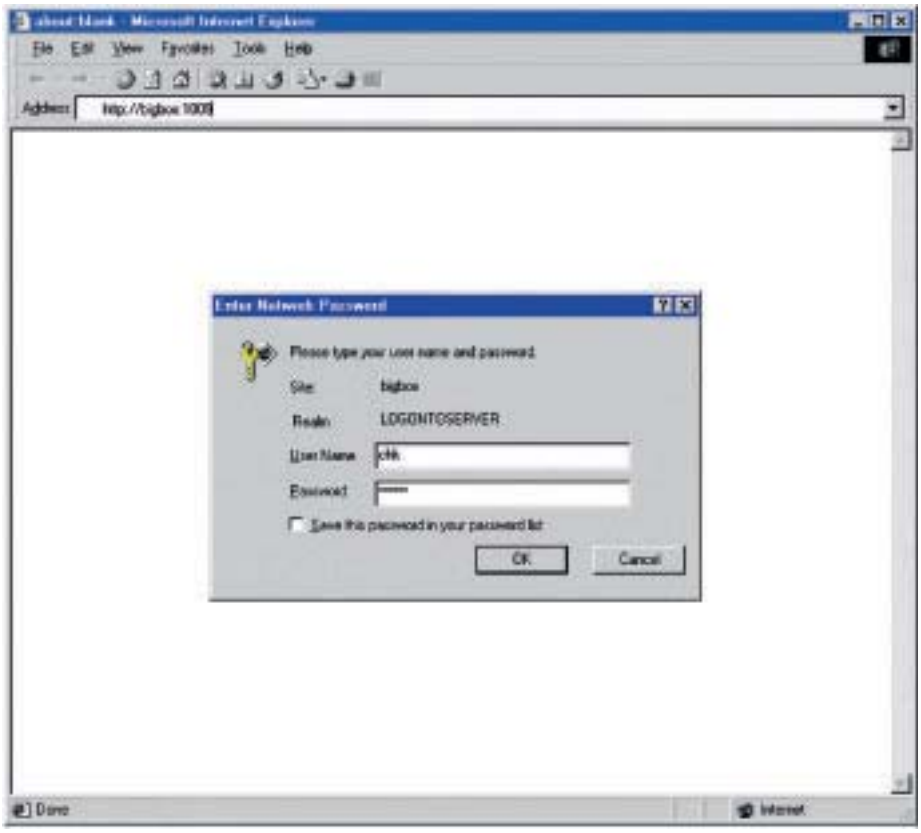
Bei dynamischen Seiten, und das dürfte der überwiegende Teil sein, kommt man nicht umhin, pro darzustellender Seite eine APL-Funktion zu schreiben, die die entsprechenden Elemente wie Eingabefelder, Selektionslisten, Radio-Buttons und Checkboxen enthält. Hier hat man den Vorteil, dass die APL-Funktion auf bestimmte Verhältnisse reagieren und die Seite dementsprechend dynamisch abwandeln kann (Beispiel: Hat der Benutzer die Berechtigung, diese Eingabe zu machen, dann binde das Eingabefeld in die Seite ein, andernfalls lasse es weg).

Benutzer-Autorisierung

Bei einer Einzelplatz-Anwendung meldet sich der Benutzer am System mit USERID und Passwort an. Erst dann wird für ihn die Anwendung gestartet und das System sorgt dafür, dass er nur die Dinge tun kann, die seiner Berechtigung entsprechen.

Bei einer Web-Anwendung muss diese Prüfung selbstverständlich ebenfalls erfolgen, wobei hier natürlich die Mainframe-Anwendung im Vordergrund steht, bei der die Autorisierung von besonderer Bedeutung ist.

Die Lösung dieses Problems heisst „FTP“. Der FTP-Server auf IBM-Mainframes gestattet es, einen Benutzer mittels RACF zu legitimieren, als ob er sich direkt am Mainframe anmelden würde. Die gängigen Web-Browser unterstützen die Anmelde-Prozedur mit einem eigenen Bildschirm-Dialog, den das Anwendungsprogramm auf dem Server bei Bedarf anfordern kann:



Die Benutzereingaben, USERID und Passwort, werden verschlüsselt an den Server geschickt, wo dann mittels FTP eine Anmeldung auf dem Mainframe versucht werden kann. Klappt diese, darf der Benutzer weiterarbeiten, klappt sie nicht, wird er vom Server abgelehnt.

Sobald der Benutzer sich einmal über diese Prozedur legitimiert hat, sendet der Browser bei jeder Transaktion diese verschlüsselte Information mit, so dass das Anwendungsprogramm jederzeit darauf zugreifen kann. Es muss allerdings gesagt werden, dass die Verschlüsselung nur den BASE64-Code verwendet, der sehr einfach zu entschlüsseln ist. Es handelt sich also nicht um eine sichere Verschlüsselung. Im Intranet, wo mit bösartigen Angriffen eigentlich nicht zu rechnen ist, dürfte das aber genügen.

Steuerung der Anwendung

Wir gehen hier von der Annahme aus, dass der Web-Server viele Benutzer bedient, aber nur eine einzige Anwendung bereitstellt. Es wäre auch machbar, dass der gleiche Server verschiedene Anwendungen betreibt, was aber den Verwaltungsaufwand um einiges erhöht und nicht notwendig ist, da ja mehrere Server gleichzeitig betrieben werden können.

Wie anfangs erwähnt, muss in die Anwendung ein „Gedächtnis“ eingebaut werden, welches für jeden Benutzer bei jeder Transaktion den Zustand wieder herstellt, den die Anwendung bei der letzten Transaktion hatte.

Um diese Aufgabe zu lösen, werden zwei verschiedene Informationsblöcke benötigt. Der erste Block ist die Statusinformation:

1. Wer stellt diese Anforderung?
2. Ist der Benutzer schon legitimiert?
3. Wo befinden sich die Daten für diesen Benutzer?

Diese Informationen müssen vorliegen, bevor der Server die eigentliche Transaktion bearbeiten kann. Sie sind von der Anwendung unabhängig.

Der zweite Informationsblock besteht aus den Daten zur Anwendung:

1. Welche Transaktionen wurden bereits durchgeführt?
2. Was wurde dabei an Daten eingegeben?
3. Welche neuen Daten sind eingegeben worden?
4. Wie wurde die vorliegende Transaktion abgeschlossen?

Dazu muss man sich Folgendes vor Augen halten:

Die Anwendung hat eine Dialogseite zum Client geschickt. Auf dieser Seite konnte der Benutzer Daten eingeben und er hat diverse Möglichkeiten, Aktionen auszulösen. In der alten Anwendung wurde dies durch die PF-Tasten bewerkstelligt, z.B. PF3: Ende, Enter: Weiter, PF1: Hilfe usw.

Im Browserfenster werden diese möglichen Aktionen durch Klick-Buttons dargestellt, da es keine PF-Tasten gibt und die Enter-Taste u.U. für andere Zwecke gebraucht wird (z.B. für den Zeilenwechsel in einem mehrzeiligen Eingabefeld). Die Klick-Buttons haben nun aber auf der Client-Seite immer die selbe Wirkung: der Inhalt der Seite wird an den Server geschickt. Das Anwendungsprogramm muss aus den übermittelten Informationen herausfinden, was als nächstes zu tun ist.

Da der Server keinerlei Kenntnisse über die Anwendung und den Benutzer hat, kann er nur eines tun, nämlich das Hauptprogramm der Anwendung aufrufen, und diesem die Daten zur Verfügung stellen, die mit der Anforderung übermittelt wurden. Und selbst der Name des Hauptprogramms muss dem Server mitgeteilt werden, ja, sogar die Tatsache, dass es sich um ein Programm handelt, und nicht um eine HTML-Datei auf der Platte.

Diese Informationen müssen bereits in der Seite enthalten sein, die der Browser gerade darstellt.

Es kann hier kein Einführungskurs in HTML mitgeliefert werden, daher werden Kenntnisse in HTML vorausgesetzt. Die erforderlichen Informationen befinden sich im „action"-Parameter des FORM-Tags der Web-Seite und sehen folgendermassen aus:

```
<FORM ... action="apl-bin/RUN_APP?logon=1&anw=xxx&daten=hugo">
```

Die Angabe „`apl-bin`" sagt dem Server, dass es sich nicht um eine Datei, sondern ein Programm handelt. Der nächste Teil „`RUN_APP`" ist der Name des Programms (das im Workspace vorhanden sein muss) und der Rest, nach dem Fragezeichen, ist der oben beschriebene erste Informationsblock.

Der Web-Server bereitet den Programm-Aufruf vor, indem er die nach dem Fragezeichen vorgefundenen Parameter als Character-String in einer APL-Variablen „`http_PARAMETERS`" bereitstellt. Den Inhalt der vom Browser zurückgeschickten Eingabefelder und sonstigen aktiven Elemente legt der Web-Server in der Variablen `http_POSTDATA` ab. Mit einer Hilfsfunktion kann man die in diesen beiden Variablen enthaltenen Definitionen in APL-Variablen mit entsprechendem Inhalt umwandeln. Die Funktion „`RUN_APP`", die vom Web-Server aufgerufen wird, muss nun für den Aufbau der korrekten Umgebung sorgen. Wie unter „Autorisierung" ausgeführt, schickt der Browser den Benutzernamen und das Passwort bei jeder Transaktion mit. Die Funktion `RUN_APP` hat also alle benötigten Informationen, um die Umgebung für die Anwendung bereitzustellen. Sie kennt den Benutzernamen, den Anwendungsnamen (`anw=xxx`) sowie den Autorisierungs-Status (`logon=1`) und eine Zusatzinformation (`daten= hugo`).

Das „Gedächtnis" der Anwendung wird mit einer DB2-Tabelle realisiert. Diese hat im Minimum vier Felder:

NAME	der Name des gespeicherten Objekts
USER	die Benutzerinformation
SEQ	eine laufende Nummer für die Eintragsverwaltung
CODE	der Inhalt des Objekts

Die Felder „NAME", „USER" und „SEQ" sind Index-Felder. Das Feld „USER" beinhaltet nicht nur die USERID des Benutzers, sonden hat intern noch zwei weitere Teilschlüssel, nämlich den Anwendungsnamen und den oben erwähnten „daten"-Teil. Ein vollständiger USER-Eintrag könnte z.B. so aussehen: '`CHK.XXX.HUGO`', wobei „HUGO" ein Name ist, den der Benutzer in einem ersten Browserdialog eingegeben hat. Auf diese Weise kann man in einer einzigen Tabelle die Daten vieler Benutzer, mehrerer Anwendungen und, für jeden Benutzer, mehrere Versionen der Daten einer Anwendung speichern, was die Verwaltung erheblich vereinfacht.

Im Feld CODE, einem LONGVARCHAR-Feld, sind dann die einzelnen Variablen-Inhalte gespeichert. Je nach der Grösse und Anzahl der Variablen müssen allerdings mehrere solcher Sätze gespeichert werden, da ein LONGVARCHAR-Feld nur rund 32k gross werden kann. Daher noch das zusätzliche Feld SEQ, welches ein Integer-Feld ist und die laufende Nummer der Folge-einträge eines Haupteintrags speichert.

Ein kleines APL-Programm zum Speichern dieser Daten, sowie ein entsprechendes zum Lesen und Generieren der APL-Variablen muss dafür vorhanden sein.

Die Anwendungs-Hauptfunktion „`RUN_APP`" liest also aus dieser Tabelle die benötigten Benutzerdaten, die, wie zu Anfang gesagt, ja den Hauptteil der Status-Informationen einer Anwendung darstellen. Es fehlt aber noch ein weiterer, wichtiger Teil: An welcher Stelle der Anwendung stand der Benutzer bevor die jetzt vorliegende Transaktion stattfand, wohin muss also die Anwendungs-Hauptfunktion verzweigen, um nahtlos weitermachen zu können?

Man könnte sich nun vorstellen, dass dies ja durch den Stand der soeben aus der Datenbank eingelesenen Variablen ersichtlich ist. Dem ist jedoch nicht so. Wer Web-Browser kennt, der weiss, dass es in jedem Browser einen „Zurück"-Button gibt. Was also, wenn der Benutzer auf diesen Knopf gedrückt hat, und somit einen vorhergehenden Dialogteil nochmals abgeschickt hat? Die bereits vorliegenden Daten der späteren Dialoge dürfen dann nicht mehr berücksichtigt werden, und es

muss ein völlig anderer Dialogteil als nächstes zum Browser geschickt werden.

Dies lässt sich durch einen recht einfachen „Trick“ bewerkstelligen. Im Allgemeinen wird ein Dialog in zwei Teilen bearbeitet. Zuerst wird der an den Benutzer zu schickende Dialog initialisiert (d.h. mit passenden Default-Werten vorbelegt), und an den Bildschirm (hier den Browser) geschickt. Wenn der Benutzer geantwortet hat, werden seine Eingaben geprüft, und dann entsprechend weiterverfahren. Die Funktion, die die HTML-Seiten generiert, setzt nun in jede Seite zwei „hidden“-Felder ein. Das sind Felder, die in der HTML-Seite zwar existieren, vom Browser aber nicht dargestellt werden. In diesen beiden Feldern, die hier „`verlauf`“ und „`pruef`“ genannt werden, ist der Zustand des Dialogs gespeichert. Jeder Dialog hat einen eindeutigen Namen. Das Feld „verlauf“ wird nun einfach von jeder Dialog-Funktion um den Namen des gerade generierten Dialogs erweitert. Beispiel:

```
<INPUT type="hidden" name="verlauf" value="bild1;bild2;bild3;">
```

Die Funktion, die den Dialog mit dem Namen „bild4“ generiert, hängt nun einfach diesen Namen an die Liste im Feld „verlauf“ hinten an. Wenn dann die Antwort vom Browser empfangen wird, kann das Hauptprogramm sehr einfach am Inhalt der Variablen verlauf feststellen, welche Dialogteile schon durchlaufen wurden.

Das Selbe geschieht mit dem Feld „`pruef`“. Wenn die Benutzereingaben eines Dialogs erfolgreich geprüft wurden, also kein Fehler festgestellt wurde, wird der Name des geprüften Dialogs an die Liste im Feld „`pruef`“ angehängt. Wenn die Antwort vom Browser zurückkommt, weiss somit die Hauptfunktion, dass die Eingaben zum vorhergehenden Dialog nicht mehr geprüft werden müssen, und kann den nächsten Dialog generieren.

Sehr oft wird jedoch noch eine weitere Information benötigt, nämlich, in welchem Feld der Cursor stand, als der Dialog verlassen wurde. Das ist schon schwieriger festzustellen, da hier die Anwendung ja selbst nicht beteiligt war, sondern der Browser diese Information liefern muss. Es muss also ein Weg gefunden werden, den Browser dazu zu veranlassen, die Information zu liefern. Auch hier bedient man sich wieder eines „hidden“-Feldes mit dem Namen „`curpos`“. Die Existenz dieses Feldes ist zwar die Voraussetzung für den Erhalt der Information, es muss aber auch vom Browser gefüllt werden. Dazu bedarf es des Einsatzes von JavaScript, im Browser muss also JavaScript aktiviert sein. Für jedes Feld im Dialog, dessen Cursor-Position zurückgemeldet werden soll, muss nun ein Event-Handler definiert werden. Dieser sieht so aus:

```
<INPUT type="text" name="feld1" onFocus="myform.curpos.value=feld1">
```

Bei diesem Beispiel wurde vorausgesetzt, dass das Formular der Web-Seite den Namen “`myform`” hat. Der Event-Handler “`onFocus`” wird automatisch aufgerufen, wenn ein aktives Element des Webseiten-Formulars den Focus erhält, also der Benutzer darauf klickt oder mit der Tab-Taste dorthin springt. Wenn die Antwort vom Browser zurückkommt, kann jetzt also die Anwendung in der Variablen “`curpos`” nachsehen, wie das Feld heisst, in dem der Cursor zuletzt war.

Die Hauptfunktion hat damit alle relevanten Informationen, um die nächste darzustellende Dialogseite zu generieren. Bevor sie diese an den Browser schickt (bzw. sie dem Web-Server zum abschicken übergibt), muss sie natürlich sämtliche relevanten Daten (= Variablen) in der DB2-Tabelle speichern.

Weiter oben wurde schon erwähnt, dass zwei Hilfsfunktionen existieren müssen, mit denen die Variableninhalte der Anwendung in der DB2-Tabelle gespeichert, und auch wieder zurückgelesen und generiert werden können.

Beim Speichern der relevanten Daten erhebt sich die Frage, was denn die relevanten Daten sind. Bei kleineren, übersichtlichen Anwendungen mag diese Frage relativ einfach zu beantworten sein. Bei grossen Anwendungen mit hunderten von Daten ist das nicht mehr gewährleistet. Selbst bei kleinen Anwendungen wird die Wartung schwieriger, wenn man bei der Einführung neuer Variablen jedesmal daran denken muss, auch die Speicherfunktion entsprechend zu ändern. Die Entscheidung liegt also nahe, einfach sämtliche Variablen des Workspaces zu speichern. Dies ist jedoch eine vorschnelle Entscheidung, denn es gibt Variablen, die auf keinen Fall gespeichert werden dürfen. Es darf nicht vergessen werden, dass es nicht mehr die Hauptfunktion der Anwendung ist, die als oberste Funktion läuft, sondern der Web-Server, der ja auch Variablen zu seiner Verwaltung benötigt. Ebenso dürfen die Variablen im oben beschriebenen ersten Infoblock, sowie USERID, Passwort und die Variablen `http_POSTDATA` und `http_PARAMETERS` nicht mitgespeichert werden, da sie ja bei jeder Transaktion neu geliefert werden und von den gespeicherten Daten nicht überschrieben werden dürfen. Es muss daher ein Mittelweg beschritten werden. Gespeichert werden alle Variablen des Workspaces, mit Ausnahme der hier erwähnten Variablen. Die Liste der Ausnahmen ist wesentlich kürzer und bleibt für sämtliche Anwendungen, auch bei deren Änderung, stabil.

Das Hilfsprogramm zur Speicherung der Variablen sollte dann, aus Sicherheitsgründen, auch sämtliche gerade gespeicherte Variablen aus dem Workspace löschen, damit sie nicht für einen anderen Benutzer sichtbar sind. Dies kann gefahrlos getan werden, da die Speicherfunktion die letzte Funktion ist, die aufgerufen wird, bevor die neue Dialogseite an den Browser geschickt wird.

Antwortzeiten

Eine Einzelplatz-Anwendung kann sich – in Grenzen – fast beliebig viel Zeit nehmen, um die Anforderung des Benutzers zu beantworten. Schlimmstenfalls gibt sie, z.B. während längeren Berechnungen, Statusmeldungen an den Bildschirm des Benutzers aus. Bei Anwendungen im Web ist das schlechterdings unmöglich. Eine Anwendung, die quasi-gleichzeitig mehrere Benutzer bedient, darf für jede Antwort nur Bruchteile von Sekunden benötigen (die Übertragungszeit im

Netzwerk steht dabei auf einem anderen Blatt). Wenn man - als Rechenbeispiel – annimmt, dass die Rechenzeit für eine Anforderung 1 sec. beträgt und der Server momentan 100 Benutzer bedient, dann wartet der zuletzt gekommene Benutzer schon fast zwei Minuten auf sein Ergebnis.

➪ Dies bedeutet, dass die Anwendung selbst kritisch auf Engpässe untersucht und optimiert werden muss.

Bei Laufzeituntersuchungen fielen z.B. die Funktionen zur Speicherung und Wiederherstellung der Anwendungsvariablen auf. Diese sind natürliche Kandidaten für die Optimierung, da sie bei jeder Transaktion einmal aufgerufen werden. Das ursprüngliche Konzept war, dass eine Liste der zu speichernden Variablen gebildet wurde, die dann der Systemfunktion `2 TF` mit „each“ übergeben wurde. Dies erwies sich bei mehreren hundert Variablen schnell als unpraktikabel, weil zu langsam. Die Optimierung bestand zunächst darin, dass eine Variable mit zwei Elementen gebildet wurde, wobei das erste Element ein String von Strings mit allen Variablen-Namen, das zweite Element ein String aller Variablen-Inhalte war. Von dieser Variablen wurde dann die Transfer-Form gebildet und gespeichert. Diese Optimierung brachte bei rund 1.000 Variablen eine Verbesserung von mehr als 80%. Allerdings gab es Probleme beim Wiederherstellen der Variablen. Es wurde ein APL-Statement als Character-String gebildet, das die Zuweisung sämtlicher Variablen-Inhalte zu ihren Namen beinhaltete. Dieser Characterstring wurde dann mittels „execute“ aktiviert. Leider funktioniert das nur bis zu einer gewissen Länge des Character-Strings. Die Aktivierung muss also in einer Schleife vorgenommen werden, wobei jedesmal nur ca. 300 Variablennamen vor der Zuweisung stehen dürfen.

Weitere Experimente führten zu einer nochmaligen Beschleunigung des Speicher-Vorgangs, wenn man statt der Transfer-Form die externe Funktion ATS (APL To SCAR) anwendet. ATS ist wiederum ca. 80% schneller als die Transfer-Form, funktioniert jedoch nur bei Variablen. Da hier jedoch nur Variablen gespeichert werden, ist diese Einschränkung gegenstandslos.

Bleibt noch ein Problem: Langdauernde Berechnungen, die sich auch durch verschärftes Tuning nicht unter die „magische“ Grenze von 1 sec. drükken lassen.

➪ Solche Anwendungsteile müssen ausgelagert und mittels eines Batch-Jobs erledigt werden.

Zum Starten dieses Batch-Jobs auf dem Grossrechner kann man FTP verwenden. Schreibt man mittels FTP eine Datei, die JCL enthält, und hat vorher den FTP-Befehl „`SITE FILETYPE=YES`“ abgesetzt, dann wird auf dem Mainframe ein Job gestartet. Dieser Job, der natürlich wieder APL startet, führt die langdauernden Berechnungen durch und kann z.B. sein Ergebnis in die DB2-Tabelle schreiben, die die Benutzervariablen enthält. Dort kann das Anwendungsprogramm leicht nachsehen, ob die benötigten Daten bereits da sind.

Benutzerbezogenes Dateihandling

Bei vielen Anwendungen werden erzeugte oder eingegebene Daten in Dateien unter dem Schlüssel des aktiven Benutzers gespeichert. Da der Benutzer seinen eigenen Adressraum hat und unter seiner USERID am System angemeldet ist, ist das ein einfacher Vorgang.

In einer Client/Server-Umgebung muss man auch hierbei neue Wege gehen, da ja nicht der Benutzer am System angemeldet ist, sondern die Server-Task. Hierfür bietet sich wieder FTP als Problemlösung an. Mittels FTP wurde ja schon die Legitimation des Benutzers überprüft und evtl ein Batch-Job gestartet, nun kann FTP seine eigentliche Stärke ausspielen: das Dateihandling. Mittels FTP wird für den Benutzer ein Logon am Zielrechner gemacht (das kann jeder Rechner sein, auf dem ein FTP-Server läuft) und dann die Daten unter der USERID des Benutzers gespeichert. Hierbei sei besonders auf ein sehr hilfreiches Feature von FTP hingewiesen: den WRITEU-Befehl. Das „U“ steht für „unique“ und bedeutet, dass man beim Speichern einer Datei immer wieder den gleichen Dateinamen angeben kann. FTP stellt fest, ob der Name bereits belegt ist und wandelt den Dateinamen gegebenenfalls so ab, dass ein neuer, eindeutiger Name entsteht. Diesen neuen Dateinamen meldet es natürlich an das aufrufende Programm zurück.

Eine weitere, sehr praktische Möglichkeit des Dateihandlings stellt HTML zur Verfügung. Mittels eines Input-Feldes vom Typ „FILE“ kann man nämlich Dateien von dem PC, auf dem der Browser läuft, an den Server schicken. Damit ist es ohne grossen Aufwand machbar, dass der Benutzer Daten in einer Datei auf seinem eigenen PC erfasst und sie dann über den Browser an die Anwendung überträgt. Bei Umlauten und Sonderzeichen muss das empfangende Programm, sofern es auf einem System mit anderem Zeichensatz (Grossrechner!) läuft, allerdings eine Zeichenumsetzung vornehmen.

Der Weg in die andere Richtung ist auch möglich, wenn auch nur auf Umwegen. Nur in den seltensten Fällen wird auf dem PC des Benutzers ein FTP-Server verfügbar sein. Wenn ja, ist das natürlich die eleganteste Lösung. Eine andere, praktikable Lösung ist, dass es einen Server im Netzwerk gibt, auf den alle Benutzer Zugriff haben und auf dem ein FTP-Server läuft. Dann können die Dateien auf diesen Server geschickt werden und der Benutzer kann sie sich von dort abholen.

Aber auch wenn diese beiden Wege nicht zur Verfügung stehen, gibt es noch eine, wenn auch nicht so elegante, weitere Möglichkeit: Die zu sendende Datei wird als reine Textseite am Browser dargestellt. Der Benutzer kann nun auf der dargestellten Seite mit der rechten Maustaste das Kontextmenü des Browsers aufrufen und dort „Quelltext ansehen“ auswählen. Diesen Quelltext kann er auf seinem PC als Datei speichern. Er muss allerdings die in diesem Text enthaltenen HTML-Tags nachträglich entfernen, was aber kein allzugrosser Aufwand ist, da nur am Anfang und am Ende der Seite einige wenige HTML-Tags nötig sind.

Eingabe von APL-Zeichen

In manchen Anwendungen wird die Eingabe von APL-Zeichen benötigt. Das war bisher natürlich kein Problem, da der Benutzer ja direkt mit einer APL-Sitzung verbunden war. Nach der Umstellung auf HTML ist das schon schwieriger, aber auch machbar. Voraussetzung ist jedoch, dass auf dem PC des

Benutzers ein IBM-APL2 für Windows installiert ist. Dies deswegen, weil der APL2 TrueType-Font zur Darstellung der Zeichen im Browser gebraucht wird, und weil der APL2-Keyboardhandler `apl2key.exe` zur Eingabe der APL-Zeichen erforderlich ist.

Das HTML-Eingabefeld muss via Style-Angabe mit dem APL2-Font belegt werden:

```
<INPUT type="text" name="hugo" style="font-family:Courier APL2;">
```

Wenn der Cursor in diesem Eingabefeld steht und `apl2key.exe` aktiv ist, kann man mit der "strg+backspace" Tastenkombination die APL-Tastatur einschalten und dann APL-Zeichen eingeben. Auf der Server-Seite muss allerdings eine Zeichenumsetzung stattfinden, damit das Anwendungsprogramm die empfangenen Zeichen auch als APL-Zeichen erkennt.

Was passiert im Fehlerfall?

Hier soll von zwei verschiedenen Typen von Fehlern gesprochen werden: den Anwender-Fehlern und den Programmfehlern. Da es selbstverständlich sehr viel mehr Anwenderfehler als Programmfehler gibt, werden zuerst die Anwenderfehler behandelt.

In den typischen 3270-Dialogmasken hatte man ja nicht sehr viele verschiedene Möglichkeiten der Benutzereingabe. Der Normalfall ist ein Eingabefeld, in das der Benutzer irgendwelche Zeichen eingeben kann. Üblicherweise ist dann ein Programm nachgeschaltet, das die Eingabe des Benutzers auf Plausibilität prüft und gegebenenfalls entsprechende Fehlermeldungen zurückschickt „Hier ist nur die Eingabe J oder N erlaubt" oder so ähnlich.

Mit den Gestaltungselementen von HTML werden viele solcher Prüfungen überflüssig. Verwendet man zur Beantwortung von Ja/Nein-Abfragen Radiobuttons oder Checkboxen, so ist überhaupt keine Prüfung der Eingabe notwendig. Ebenso entfällt die Prüfung der Eingabe, wenn der Benutzer nur Optionen aus einer Drop-Down-Liste auswählen kann. Selbst sogenannte Crosschecks (wenn hier „Ja" ausgewählt wurde, kann dort nicht Option 4 gewählt werden) lassen sich z.T. sehr elegant mit ein paar wenigen JavaScript-Zeilen direkt „vor Ort" erledigen.

Die Prüfung der Eingaben in den normalen Eingabefeldern bleibt wie sie war. Muss eine Fehlermeldung an den Benutzer geschickt werden, dann muss die gesamte Seite, mit den vom Benutzer eingegebenen Daten und einem Fehlertext an geeigneter Stelle zurückgeschickt werden.

Programmfehler sind die andere Seite der Medaille. Natürlich schreiben wir praktisch fehlerlosen Code, aber dieser andere Programmierer da, der die Anwendung vor uns betreut hat...

Es versteht sich von selbst, dass der Server nicht wegen eines Fehlers im Anwendungsprogramm abstürzen darf. Der Web-Server in dem bereits erwähnten Programmpaket Web4APL führt daher das Anwendungsprogramm normalerweise unter „Execute Controlled" (`/EC`) aus. Passiert ein Fehler, wird eine HTML-Seite mit der Fehlermeldung von APL an den Browser geschickt.

Dies ist für den Produktionsbetrieb die richtige Wahl, allerdings nicht für den Test eines Programms. Man darf nicht vergessen, dass die Anwendung nach der Umstellung auf Client/Server und HTML nur noch über den Server aufgerufen werden kann, da ja alle Benutzerinteraktion über den Browser abgewickelt wird. Wenn die Anwendung unter ðEC abläuft, kann man das Programm natürlich auch nicht zu Testzwecken an einer Stelle anhalten, da der Stopvektor dann ignoriert wird. Daher muss der Server eine Möglichkeit enthalten, das Anwendungsprogramm zum Test statt unter ðEC mit normalem „Execute" aufzurufen.

Hat man nun den Server zur Diagnose eines Fehlers angehalten, so wartet der Browser weiter auf die eigentlich fällige nächste Seite. Irgendwann gibt der Browser das Warten auf und meldet: „Server nicht gefunden". Jetzt zahlt sich das Konzept aus, dass jede HTML-Seite alle benötigten Informationen für die Steuerung der Anwendung in sich trägt. Man kann nämlich nach der Fehlerbehebung den Server einfach neu starten und im Browser auf den „Zurück"-Knopf klicken. Die vorherige Seite wird angezeigt, man klickt auf „Weiter" (oder wie das Pendant zur früheren Eingabetaste heisst) und die Anwendung läuft an der unterbrochenen Stelle wieder an.

Fazit

Die Umstellung einer Anwendung auf moderne Web-Technologie ist zwar mit einigem Aufwand verbunden, bietet aber deutliche Vorteile für den Anwender, den Programmierer und die Firma.

Der Anwender kann mit den heute üblichen und gewohnten GUI-Eingabedialogen arbeiten und in vielen Fällen mit der Maus, statt mit der Tastatur arbeiten. Die Dialoge können sehr viel übersichtlicher und intuitiver gestaltet werden, und die Beschränkung auf 24x80 Zeichen fällt weg, so dass zum Teil auch Dialoge, die bisher aus Platzgründen auf mehrere Masken verteilt waren, zusammengefasst werden können.

Der Programmierer kann sich mehr auf die eigentliche Anwendungs-Funktionalität konzentrieren und ist nicht mehr überwiegend damit beschäftigt, Prüffunktionen zu schreiben. Auch die Fragen von Benutzern werden weniger, da die Dialoge leichter verständlich sind.

Für die Firma ergeben sich noch weitere Vorteile, denn die Anwendung kann jetzt z.B. auch über das Internet erreicht werden, da die Terminal-Emulation nicht mehr benötigt wird. Ausserdem hat eine im Web-Browser laufende Anwendung einen ganz anderen Image-Stellenwert als eine 3270-basierte Anwendung in der Terminal-Emulation.

Kontakt:

Conrad Hoesle-Kienzlen, Augustenstr. 41, D-70178 Stuttgart, eMail: chk@hoesle-kienzlen.de, Internet: www.hoesle-kienzlen.de

Herbert Voß

LyX

Open Source Document Processor

LYX als sogenanntes Frontend zu LATEX, welches wiederum ein Frontend zu TEX ist, versucht etwas zu optimieren, was LATEX selbst gar nicht leisten will: eine hochwertiges Satzsystem wie TEX einem an tiefsinnigen typographischen Fragen relativ uninteressierten Benutzer zugänglich zu machen. LYX versteht sich daher auch nicht als Textprocessor, sondern als Document Processor, denn es verfügt selbst über keinerlei eigene Formatierungsmöglichkeiten. Alle beschriebenen Eigenschaften von LYX beziehen sich auf die derzeit offizielle Version LYX 1.1.6fix2. Eine erheblich erweiterte 1.2 Version ist für den Herbst zu erwarten.

Die Geschichte

Unter Federführung von Donald Knuth[1] wurde ausgehend von den Prinzipien des herkömmlichen Drucksatzes ein System entwickelt, welches Anwendern einen professionellen Satz von wissenschaftlichen Publikationen ermöglichte. In der ersten Version nannte Knuth dieses System tec, sehr bald nur noch TEX geschrieben, aber immer „Tech“ gesprochen. 1977 erschien die erste veröffentlichte Version von TEX, 1982 die zweite Version und 1990 die dritte und letzte Version.

Eine weise Entscheidung traf Knuth indem er beschloss, TEX *nicht* weiterzuentwickeln! Das Problem in der Softwareentwicklung ist keinesfalls eine in ihren Eigenschaften begrenzte Anwendung zu entwickeln, sondern zu erkennen, dass man sich mit seiner Entwicklung auf dem Weg der „Eierlegenden Wollmilchsau“ befindet, der sogenannten universellen Anwendung. Diese fordert ab einem bestimmten Punkt einen immer größer werdenden Entwicklungs- und Wartungsaufwand bei einem immer kleiner werdenden Fortschritt in der Anwendung. TEX kann auch als erstes wegweisendes Open-Source-Projekt verstanden werden, denn Knuth stellte von Anfang an den gesamten, in der Sprache WEB erstellten Quellcode von TEX zur Verfügung. Dies führte dazu, dass TEX faktisch für jedes System angepasst wurde. TEX in seiner Ursprungsversion kann nicht nur aus heutiger Sicht als kryptisch bezeichnet werden, was Leslie Lamport[2] bereits Anfang 1980 veranlasste, eine Sammlung von Makropaketen zu entwickeln, welche als LATEX 2e den heutigen aktuellen Stand darstellen.[3] Daneben gibt es eine Vielzahl von anderen Entwicklungen, die alle auf TEX basieren, hier aber nicht weiter erwähnt werden sollen. Information gibt es reichlich im Comprehensive TEX Archiv Network (CTAN)[3]. Seit einigen Jahren wird intensiv an einer Fortentwicklung gearbeitet, die un-

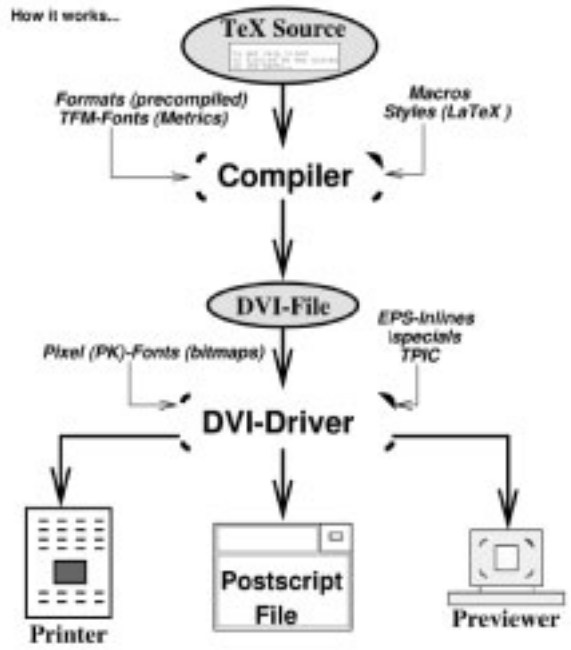


Abbildung 1: Der formale Aufbau eines TEX-Systems (http://www.rhrk.uni-kl.de/~knopper/tex/howitworks.gif)

1 http://www-cs-faculty.stanford.edu/~knuth/
2 http://www.research.compaq.com/SRC/personal/lamport/
3 http://www.ctan.org. Deutscher mirror: http://www.dante.de

ter dem Projektnamen LaTeX3 läuft, während Lamport selbst die Arbeit an LaTeX eingestellt hat. Den formalen Aufbau eines TeX-Systems zeigt Abb. 1, aus der deutlich zu erkennen ist, dass LaTeX nur eine untergeordnete Bedeutung in diesem Gesamtsystem spielt. Das Fehlen einer graphischen Benutzeroberfläche für LaTeX kann grundsätzlich als unwesentlich betrachtet werden, stellte aber in der heutigen „Alles-so-schön-bunt-Welt“ ein wesentliches Kriterium dar, um 1994/95 im Rahmen einer Diplomarbeit eine grafische Oberfläche zu entwickeln, die in Anlehnung an „Lyrik“ über „LyriX“ nun LyX genannt wird. Ziel der Entwicklung war und ist es immer noch, eine Umgebung für das in Abb. 1 dargestellte TeX-System zu entwickeln, die den Anwender auch von eingehenden LaTeX-Kenntnissen befreien sollte. Bleibt also festzustellen, dass man eigentlich *immer* von TeX reden müsste, denn alle anderen Programmumgebungen sind nur aufgesetzt.

Für das Verständnis der Philosophie von LyX ist dies jedoch unerheblich, insbesondere für den Nutzer, der LyX gerade wegen der gewissen TeX- bzw. LaTeX-Ferne anwenden möchte. Neben LyX gibt es noch verschiedene andere grafische Oberflächen für LaTeX, wie z.B. ε*:doc*[4].

Wysiwyg versus Wysiwym

Ein Schlagwort der 90-Jahre in der Softwareentwicklung war das Wysiwyg-Prinzip[5], was dem Anwender eine schnelle Kontrolle seiner Arbeit ermöglichen sollte. Einerseits stellt diese Forderung erhebliche Probleme an die Softwareentwicklung. Andererseits ist festzustellen, dass im Bereich der wissenschaftlichtechnischen Publikationen ein wesentlicher Teil des Layouts durch DIN-Normen oder Verlagsvorgaben festliegt, der Autor hat sich in diesem Rahmen zu bewegen. Daher entschied man sich bei der Entwicklung von LyX das Wysiwym-Prinzip[6] anzuwenden, d.h. der Anwender hat nur eine globale Ähnlichkeit zwischen dem was die grafische Oberfläche anzeigt und der späteren Ausgabe. Dabei hängt die Ähnlichkeit von der Komplexität des Textes ab, sie liegt im Bereich von 30 - 95 %, was exemplarisch durch Abb. 2 gezeigt wird.

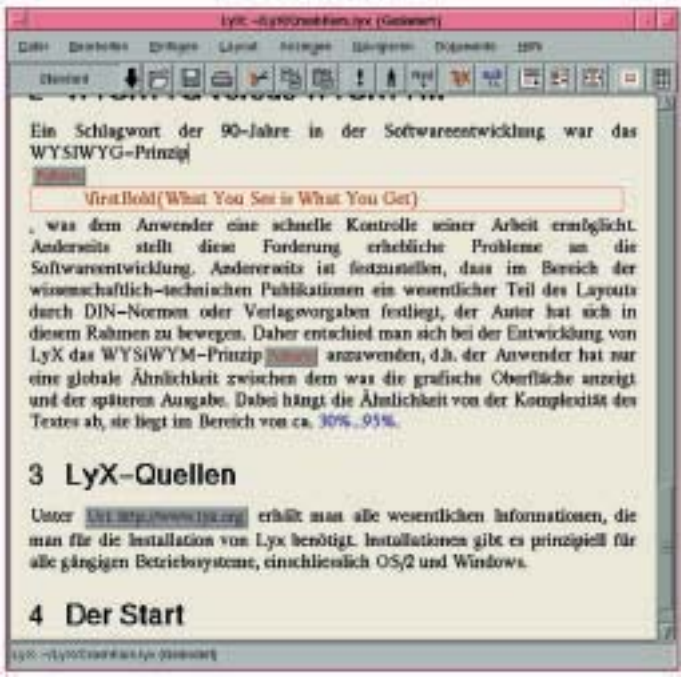


Abbildung 2: Die LyX-Oberfläche

Damit erst gar kein Missverständnis auftritt, soll noch einmal klar gestellt werden, dassWysiwym ein Kompromiss ist und dass gegen die Kombination von TeX und Wysiwyg nichts aber auch gar nichts spricht.

Warum LyX und nicht ???

Es gibt hier keine überzeugende Antwort, wenn man dem typographischen Layout nicht sonderlich viel Aufmerksamkeit schenkt. So ist z.B. der Formelsatz in WinWord überhaupt nicht mit dem von LaTeX zu vergleichen, was die folgenden Abbildungen verdeutlichen sollen. Dabei ist es hier im Prinzip unerheblich, dass die Abb. 4 als Bitmap eingefügt wurde. Der wesentliche Unterschied liegt zum einen in den verwendeten Fonts, denen man unter TeX weitaus mehr Aufmerksamkeit bei der Erstellung gewidmet hat als den bekannten TrueType-Fonts unter Windows.[7] Unter Linux wird sehr häufig das Softwarepaket StarOffice von SUN[8] empfohlen, da es zum einen kostenlos und zum anderen professionell aufgebaut ist. Im Gegensatz zu LaTeX kann es als echtes Office-Paket mit Textverarbeitung, Tabellenkalkulation, Datenbank, usw. gelten, während LaTeX bekanntermassen nur eins kann: Textverarbeitung und -bearbeitung. Dies allerdings so gut wie wohl kein anderes Softwarepaket. Jede Textverarbeitung hat bekanntermaßen ihre Stärken und Schwächen, so dass hierauf nicht weiter eingegangen zu werden braucht. Die Stärken von LaTeX liegen zweifelsohne im Bereich der technisch-naturwissenschaftlichen Literatur. Mittlerweile sind der Anwendung von LaTeX durch die Vielzahl an kostenlos zur Verfügung stehenden Paketen praktisch keine Grenzen mehr gesetzt. Am Schluss werden einige Beispiele dazu angegeben. Der Umfang eines kompletten installierten TeX/LyX-Pakets steht zwar einer Installation eines Office-Pakets in nichts nach, dennoch ergibt sich ein extremer Geschwindigkeitsvorteil für TeX bei der Formatierung von großen Dokumenten.

LyX-Layout

Das für den LyX-Anfänger größte Problem liegt sehr häufig in der Erkenntnis, dass das Layout des LyX-Fensters überhaupt nichts mit dem späteren LaTeX-Layout zu tun haben muss. Innerhalb der Wysiwym-Philosophie war man dennoch bestrebt, viele Ähnlichkeiten zum späteren Endlayout zu erreichen. Die LyX-Layouts legen lediglich fest, wie etwas auf dem Bildschirm erscheinen soll und welches das dazugehörige LaTeX-Layout ist. Für das Absatzlayout *Title* sieht das entsprechende LyX-Layout wie folgt aus (vergleiche Listing 1).

Hierbei stellt `LatexName title` den direkten Zusammenhang zu Latex her, denn dadurch wird festgelegt, was mit dem Text passieren soll, der im LyX-Fenster das Absatzlayout title aufweist (vgl. Abb. 5). Anderseits kann man seine eigenen Layouts schreiben und umfangreichen LaTeX-

4 http://members.magnet.at/hfbuch/edoc/
5 WHAT YOU SEE iS WHAT YOU GET
6 WHAT YOU SEE iS WHAT YOU MEAN
7 Auch TrueType-Fonts können unter LaTeX verwendet werden, wodurch sie allerdings auch nicht besser werden (http://www.radamir.com/tex/ttf-tex.htm).
8 http://www.sun.com/products/staroffice/

Code dabei berücksichtigen, wie das folgende Layout (siehe Listing 2) zeigt, welches ein Absatzlayout mit einem grauen Hintergrund definiert.

Listing 1: Titel-Layout

```
 1 # LyX Title-Layout
 2 Style Title
 3   Margin          Static
 4   LatexType       Command
 5   InTitle         1
 6   LatexName       title
 7   ParSkip         0.4
 8   ItemSep         0
 9   TopSep          0
10   BottomSep       1
11   ParSep          1
12   Align           Center
13   AlignPossible   Center
14   LabelType       No_Label
15 # standard font definition
16   Font
17   Family          Sans
18   Series          Bold
19   Size            Largest
20   EndFont
21 End #title
```

Listing 2: Color-Umgebung

```
 1 #
 2 # Standard color-style definition
 3 #
 4 Style myStandardColor
 5   Margin              Static
 6   LatexType           Environment
 7   LatexName           cminipage
 8   ParIndent           MM
 9   ParSkip             0.4
10   Align               Block
11   AlignPossible       Block, Left, Right, Center
12   LabelType           No_Label
13   # standard font definition
14   Font
15     color             blue
16     family            typewriter
17   EndFont
18   Preamble
19     \usepackage{color,calc}
20   \definecolor{myColor}{rgb}{0.9,0.9,0.9}% rgb color model
21     \newenvironment{cminipage}{%
22      \noindent%
23      \begin{lrbox}{\@tempboxa}%
24      \begin{minipage}{\linewidth-2\fboxsep}%
25       \begin{list}{}{% LyXcode
26        \setlength{\rightmargin}{\leftmargin}
27       \raggedright
28       \setlength{\itemsep}{0pt}
29       \setlength{\parsep}{0pt}
30      \ttfamily}%
31      \item[]}%
32     {\end{list}%
33    \end{minipage}%
34   \end{lrbox}%
35    \fcolorbox{myColor}{myColor}{\usebox{\@tempboxa}}%
36   }%
37   EndPreamble
38 End # myStandardColor
```

Intern wird der LyX-Layoutname *Standard-Color* verwendet, der dem ebenfalls selbstdefinierten LaTeX-Layout *cminipage* entspricht.

$$
\begin{aligned}
A = \lim_{n\to\infty} \Delta x \Big(& a^2 + \left(a^2 + 2a\Delta x + (\Delta x)^2\right) \\
& + \left(a^2 + 2\cdot 2a\Delta x + 2^2(\Delta x)^2\right) \\
& + \left(a^2 + 2\cdot 3a\Delta x + 3^2(\Delta x)^2\right) \\
& + \ldots \\
& + \left(a^2 + 2\cdot(n-1)a\Delta x + (n-1)^2(\Delta x)^2\right)\Big) \\
& = \frac{1}{3}\left(b^3 - a^3\right)
\end{aligned}
$$

Abbildung 3: LyX-Formelsatz

$$
\begin{aligned}
A = \lim_{n\to\infty} \Delta x \Big(& a^2 + \left(a^2 + 2a\Delta x + (\Delta x)^2\right) \\
& + \left(a^2 + 2\cdot 2a\Delta x + 2^2(\Delta x)^2\right) \\
& + \left(a^2 + 2\cdot 3a\Delta x + 3^2(\Delta x)^2\right) \\
& + \ldots \\
& + \left(a^2 + 2\cdot(n-1)a\Delta x + (n-1)^2(\Delta x)^2\right)\Big) \\
& = \frac{1}{3}\left(b^3 - a^3\right)
\end{aligned}
$$

Abbildung 4: WinWord-Formelsatz

Der Start

Die grafische Oberfläche zeigt sich für heutige Verhältnisse eher spartanisch in ihrer Aufmachung (Abbildung 2), enthält aber alle wesentlichen Elemente, die mehr oder weniger zu jeder Textverarbeitung gehören und in Tabelle 1 zusammengefasst sind. Dem Benutzer sind hierbei keine Einschränkungen vorgegeben, denn er kann diese Leiste nach Belieben seinen eigenen Vorstellungen anpassen. Einige zusätzliche Icons sind bereits in dem Standardsystem dabei, können aber auch selbst erstellt und natürlich der LyX-Community zur Verfügung gestellt werden (siehe Anhang).

LyX hat keine eigenen Formatierungseigenschaften, d.h. baut vollständig auf LaTeX. Somit kann LyX auch nur seine Arbeit beginnen, wenn als Grundvoraussetzung eine LaTeX-Dokumentenklasse ausgewählt wurde. Die Zahl der zur Verfügung stehenden Klassen hält sich in Grenzen, da in diesen nur die grundsätzlichen Dinge festgelegt werden. Im wesentlichen gibt es folgende Klassen:

- article
  - article (AMS)[9]
  - article (Komascript)[10]
  - article (IEEEtran)[11]
  - article (Springer)[12]
  - article (exam)[13]
  - article (Hebrew)

9 AMERICANMATH SOCIETY
10 Eine auf die speziellen europäischen Bedürfnisse ausgerichtete Textklasse
11 INSTITUT OF ELECTRICAL AND ELECTRONICAL ENGINEERS
12 Zeitschriften des wissenschaftlichen Springerverlags
13 Für allgemeine Diplomarbeiten

- ● book
- – book (AMS)
- – book (Komascript)
- ● report
- – report (Komascript)
- ● letter
- – letter (g-Brief) german
- – letter (g-Brief) french
- – letter (german)
- – letter (Komascript)
- ● seminar
- ● slide
- ● APA style[14]
- ● CD-Box Cover[15]

Standardmäßig wählt LyX die *article*-Klasse, wenn nichts anderes angegeben wird. Grundsätzlich kann sich jeder Anwender seine eigenen Klassen schreiben, da LaTeX als offenes System praktisch keine Grenzen hat.[1, 4] So haben viele Universitäten ihre eigen Klassen für Dissertationen[16] oder einzelne Institute für ihre Diplomarbeiten. Will man eine derartige neue LaTeX-Klasse für LyX zur Verfügung stellen, so sollte zuerst ein entsprechendes LyX-Layout (vgl. Fußnote 8) erstellt werden.

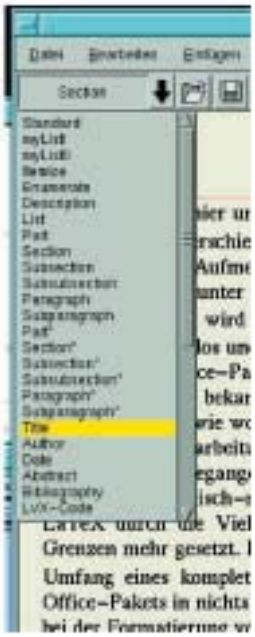


Abbildung 5: Das LyX-Layout-Menü

Symbol	Bedeutung
	Öffne Dokument
	Speichern
	Drucken
	Ausschneiden
	Kopieren
	Einfügen
	Italic
	Kapitälchen
Font	frei belegbar
TeX	TeX mode
a+b	Mathematischer Modus
	Fußnote einfügen
	Randnotiz einfügen
	Umgebungstiefe ändern
	Grafik einfügen
	Tabelle einfügen

Tabelle 1: Bedeutung der Buttons

Textparameter

Wie bei jeder Textverarbeitung sind auch bei LaTeX-Anwendungen global wirksame Parameter festzulegen, von denen hier nur einige exemplarisch angegeben werden:

- ● ein/zweispaltig
- ● Papiergröße
- ● Seitenränder
- ● Kopfzeilen
- ● Hauptsprache des Dokuments
- ● Art der Anführungsstriche

Durch die Verwendung von Postscript als Standardausgabeformat kann die Druckvorlage auch noch nachträglich in jeder beliebigen Weise verändert werden. So kann z.B. ein erstelltes Buch-Cover von der genormten Größe 37.1cm x 23cm für eine Kontrollausgabe mit dem Befehl:

```
psresize -W37.1cm -H23cm -pa4 Cover.ps | lpr
```

in DIN-A4-Format auf dem Drucker ausgegeben werden.

Der Formelgenerator

Es wurde bereits auf die hervorragende Qualität des Formelsatzes von TeX verwiesen. LyX unterstützt dies in einem graphischen Modus, der nahezu Wysiwyg Qualitäten erreicht (Abb. 6).

$$a = arg(z) = \begin{cases} 0 & (a < 0 \wedge b = 0) \\ \pi & (a < 0 \wedge b = 0) \\ \arctan \frac{Im(z)}{Re(z)} & (a \in \Re \wedge b > 0) \\ \arctan \frac{Im(z)}{Re(z)} & (a \in \Re \wedge b < 0) \end{cases}$$

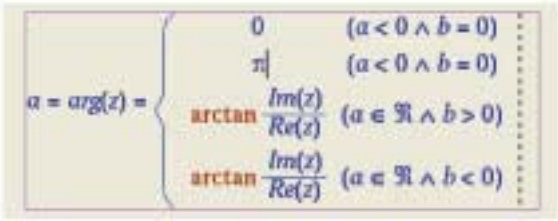


Abbildung 6: LyX im Formelmodus

Im Gegensatz zu dem ebenfalls relativ komfortablen Mathematikeditor von WinWord existieren unter LaTeX wesentlich mehr vertikale und horizontale Möglichkeiten der Ausrichtung, wohingegenWinWord im Gegensatz zu LyX das bessere Auswahlmenü im Editor hat. Das LyX-Mathpanel hat einen mehr als spartanischen Charakter, wie Abb. 7 hinreichend verdeutlicht.

---

14 AMERICAN PSYCHOLOGICAL ASSOCIATION
15 Beruht prinzipiel auf der article-Klasse
16 Vor allem in den USA üblich.
17 Unterstützt von der AMERICANMATH SOCIETY.
18 Siehe auch http://www.perce.de/lyx/Equations.pdf

Abbildung 7: LyX Math-Panel

Das *amsmath*-Paket[17] bietet umfangreiche Funktionen in Zusammenhang mit dem Formelsatz an, von denen LyX in der kommenden Version wesentliche unterstützen wird.[18]

Unterschieden wird immer zwischen einem Inline-Mode und einer abgesetzten Formel (Displaymath). Im ersten Fall ist der gesamte mathematische Ausdruck Teil einer regulären Zeile, im zweiten Fall ein eigener Absatz mit der Möglichkeit der Gleichungsnumerierung. Dabei ist einWechsel zwischen diesen beiden Modi für einen mathematischen Ausdruck per Tastenbefehl möglich, wobei automatisch eine Anpassung der Anordnung und Größe von Indizes und Exponenten erfolgt.

Als Beispiel betrachte man den folgenden mathematischen Ausdruck $\left(\prod_{i=1}^{n-1} e^{-iq2} = e^{-\frac{n(n-1)}{2}q2}\right)$, der einmal direkt in dieser Zeile und zum anderen als abgesetzte Formel geschrieben wurde.

$$\prod_{i=1}^{n-1} e^{-iq_2} = e^{-\frac{n(n-1)}{2}q_2} \qquad (1)$$

Fließobjekte (Floats)

Per Definition können Abbildungen und Tabellen in sogenannte Fließobjekte eingebunden werden, die dann von TEX in günstige Postionen gesetzt werden. Dies unterliegt gewissen Kriterien, wie z.B. minimaler Textanteil auf einer Seite, maximale Zahl von Floats auf einer Seite, Abstand zwischen Floats auf einer Seite oder Abstand zwischen Float und Text. Es sind in jedem Fall erheblich mehr Steuerungsmöglichkeiten vorhanden als sonst bei Textverarbeitungen üblich. Probleme gibt es lediglich im Spaltenmodus, wenn mehr als zwei Spalten benötigt werden. In diesem Fall muss man sich eigene „Float"- Umgebungen definieren. Zu Problemen kommt es immer dann, wenn man sehr viele Fließumgebungen innerhalb eines Kapitels hat, denn TEX setzt erst den gesamten Text und versucht dann die fließenden Objekte möglichst sinnvoll einzufügen. Dazu müssen diese natürlich erst einmal zwischengespeichert werden, was zu Speicherproblemen führen kann. Hier ist es dann notwendig manuell einzugreifen.

Neben den standardmäßig vorgegeben Objekttypen Bilder/Tabellen gibt es unter LyX noch den Algorithmus als vordefiniertes Fließobjekt. Dem Anwender sind jedoch keine Grenzen in der eigenen Definition neuer Objekte gesetzt. Für sämtliche Fließobjekte existiert ein *Listof*...-Befehl, der ein Verzeichnis gleicher Fließobjekttypen an beliebiger Stelle des Textes ausgeben kann.

Bilder und Tabellen

In der Regel befinden sich diese innerhalb von Floats, denn ihre Anordnung beinflusst wesentlich das Layout des Dokuments. Grundsätzlich kann man sie auch unabhängig davon in den Text einfügen. Sämtliche Grafiken müssen entweder das Postscript Format (Suffix „ps")- oder das Encapsulated Postscript Format (Suffix „eps") aufweisen. Da dieses Format grundsätzlich völlig ungeeignet für „real pictures" ist, sind die Dateien schon mal mehrere Megabyte groß. Es empfiehlt sich daher mit den „gezippen" Dateiformaten zu arbeiten, was allerdings voraussetzt, dass man eine weitere Hilfsdatei von wenigen Byte erzeugt, die die Werte der sogenannten Bounding Box enthält, z.B.:

```
%%BoundingBox: 0 0 376 166
```

Damit ist LATEX in der Lage ohne Einfügen des Bildes den Formatierungsprozess durchzuführen und erst nach völliger Fertigstellung des Dokuments das Bild einzufügen. Das „entzippen" der Grafik wird automatisch vom Paket *graphicx* mithilfe des Programms *gzip* durchgeführt.
Für das Einfügen von Tabellen gibt es mehrere umfangreiche Pakete, die eine Erweiterung der standardmäßigen Unterstützung von LATEX vornehmen.

LfdNr	Funktion $f(x)$	Ableitung $f'(x)$
1	$y = \int x \cdot dx$	$\frac{dy}{dx} = x$
2	Parabel	Gerade

Verweise

Diese sind auf vielfältige Art möglich, unterscheiden sich aber nicht prinzipiell von denen anderer Textverarbeitungsprogramme, so dass hier darauf nicht weiter eingegangen werden muss. Das LATEX-Paket *hyperref* unterstützt die Markierung und Hervorhebung derartiger Verweise als Link für die Umwandlung in eine pdf-Datei. Dieses muss in der derzeitigen Version noch über die LATEX-Präambel (siehe S. 38) eingebunden werden, denn LyX unterstützt momentan nur das *prettyref* -Paket, so dass man per Menü die Art

Abbildung 8: Menü zum Eintragen eines Verweises

des Querverweises bestimmen kann (Abb. 8). Für die folgende Version 1.2 ist jedoch eine vollständige Integration des *hyperref*-Pakets vorgesehen.

Verweise beziehen sich grundsätzlich auf sogenannte referenzierbare Label, die an jeder beliebigen Stelle des Dokuments mit einem frei wählbaren Namen eingetragen werden können. Da dies bei umfangreichen Texten ohne weiteres mehr als 100 Labels sein können, bietet LyX hier eine Unterstützung, indem es z.B. für Überschriftenlabels automatisch den Typ, z.B. „sec:“ für Section, vor die eigentliche Marke setzt.

LyX kann die Label nach ihrem Erscheinen im laufenden Text oder aber sortiert anzeigen. Auch Verweise auf Fußnoten sind kein Problem, müssen jedoch analog zu LaTeX mit einem kleinen Trick „erzwungen“ werden.[19]

Minipages

Diese bilden innerhalb von LaTeX eine wesentliche Voraussetzung für diffizile Layoutfragen und werden daher auch von LyX direkt unterstützt. Die Minipages, als „Seite in einer Seite“ sind prinzipiell eigenständige Dokumente mit all den Eigenschaften, die eben für einzelne Seiten gelten. Andererseits liegt damit fest, dass eine Minipage nicht größer als eine Seite werden kann und auch vollständig in dieser liegen muss, also keinen Seitenumbruch aufweisen kann.

Typische Anwendungsfälle für Minipages sind das Nebeneinandersetzen zweier Objekte (Abbildungen, Tabellen, Algorithmen, etc.) mit eigenen, fortlaufend numerierten Bildunterschriften in einer Fließumgebung.

Dies[1] soll jetzt an einem willkürlichen Beispiel gezeigt werden, bei dem dieser Text wiederholt in der Minipage[2] erscheint, wobei eine Breite von 45% für die linke und 40% für die rechte Minipage vorgesehen wurden. Auch das Einfügen von Formeln ist kein Problem:

$$f(n) = \prod_{j=1}^{n-1} e^{-iq_2} \tag{2}$$

Und Fußnoten sind auch möglich!

1 Initialen lassen sich leicht mit dem Paket-dropping erstellen.
2 Eine Minipage kann nicht größer als eine Seite selbst sein und darf keinen Seitenumbruch aufweisen!

Dies[1] soll jetzt an einem willkürlichen Beispiel gezeigt werden, bei dem dieser Text wiederholt in der Minipage[2] erscheint, wobei eine Breite von 45% für die linke und 40% für die rechte Minipage vorgesehen wurden. Auch das Einfügen von Formeln ist kein Problem:

$$f(n) = \prod_{i=1}^{n-1} e^{-iq_2} \tag{3}$$

Und Fußnoten sind auch möglich!

1 Initialen lassen sich leicht mit dem Paket-dropping erstellen.
2 Eine Minipage kann nicht größer als eine Seite selbst sein und darf keinen Seitenumbruch aufweisen!

Eingerichtet werden Minipages über Layout ⇨ Absatzformat ⇨ Extras, was sich allerdings in der neuen LyX-Version 1.2 etwas ändern wird, so dass weitere Optionen im Zusammenhang mit Minipages erleichtert werden. Abb. 9 zeigt die in der aktuellen Version möglichen Einstellungen

Abbildung 9: Absatzlayout für Minipages

Dies[1] soll jetzt an einem willkürlichen Beispiel gezeigt werden, bei dem dieser Text wiederholt in der Minipage[2] erscheint, wobei eine Breite von 45% für die linke und 40% für die rechte Minipage vorgesehen wurden. Auch das Einfügen von Formeln ist kein Problem:

$$f(n) = \prod_{j=1}^{n-1} e^{-iq_2} \tag{3}$$

Und Fußnoten sind auch möglich!

1 Initialen lassen sich leicht mit dem Paket-dropping erstellen.
2 Eine Minipage kann nicht größer als eine Seite selbst sein und darf keinen Seitenumbruch aufweisen!

von Minipages. In dem hier vorliegenden Beispiel wurde ein `\hfill` zwischen einzelne Absätze eingefügt, so dass die linke Minipage linksbündig und die rechte eben rechtsbündig erscheint.

Eine Minipage kann natürlich auch beliebig rotiert werden, was durch den LaTeX-Befehl `rotatebox` unterstützt wird. Weitere Möglichkeiten, insbesondere hinsichtlich der vertikalen Positionierung von Minipages kann man KLÖCKL[1] und KOPKA[2] entnehmen.

Verteilte Dokumente

Das Schreiben von Büchern kann sehr schnell eine Frage der Übersicht werden, wenn mehrere hundert Seiten zu organisieren sind. Hier bietet LyX das Arbeiten mit verteilten Dokumenten an, so dass das gesamte Dokument in beliebig kleine Einheiten unterteilt werden kann.

Der Aufbau des Hauptdokuments könnte dann entsprechend Abb. 10 aussehen. Es enthält nur die Angaben über die einzufügenden Kapitel und Verzeichnisse (Inhalt, Index), sowie Informationen über die Art der Seitenzahlen, die im Bereich \frontmatter (Titel, Vorwort,...) eine römische Seitennumerierung, im Bereich \mainmatter und \backmatter eine arabische Nummering aufweisen. Das Hauptdokument muss selbst also keinerlei Text aufweisen.

LyX öffnet spätestens bei dem ersten Übersetzungsvorgangs des Dokuments automatisch alle benötigten Dateien. Dies geschieht aber auch schon dann, wenn man versucht einen Querverweis von einem Teildokument in ein anderes einzufügen, denn LyX erstellt aus allen verteilten Dokumenten ein gemeinsames Labelverzeichnis, so dass Querweise zwischen verteilten Dokumenten keinerlei Probleme bereiten.

19 http://www.educat.hu-berlin.de/~voss/lyx/footnotes.html#cross

Abbildung 10: Aufbau des Hauptdokuments

Bibliographie

Dies stellt wieder einen wichtigen Schwerpunkt von LaTeX dar, denn es existieren derartig viele Softwarepakete, die alle möglichen Variationen und spezielle Sonderwünsche unterstützen. Das völlig eigenständig ablaufende Programm BibTeX setzt prinzipiell keine Grenzen in der Form der Zitate, da diese durch ein eigenes Stylefile festgelegt sind, welches vom Anwender beliebig erweitert werden kann. LyX überwacht den gesamten Mechanismus und startet bei Bedarf von sich aus das BibTeX-Programm, der Anwender kann sich um andere Dinge kümmern. Das Einfügen der Literaturzitate erfolgt über ein Menü, welches sämtliche in einer angegebenen Literaturdatenbank oder im Text durch ein Schlüsselwort festgelegte Literaturangaben enthält (Abb. 11).

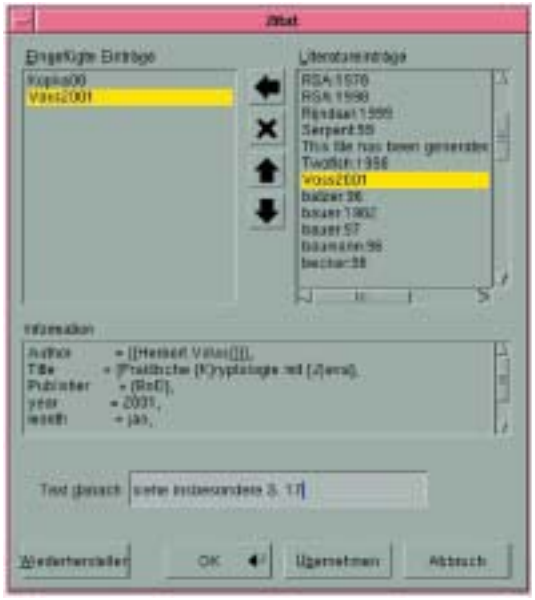


Abbildung 11: Das Menü zum Einfügen von Literaturzitaten

Grundsätzlich kann man sich eine einzige Literaturdatenbank anlegen, in die *sämtliche* Literatur wie Bücher, Zeitschriftenartikel, Tagungsberichte, Technische Berichte, unveröffentlichte Berichte und Sammlungsbände eingetragen werden kann, wobei der Aufbau eines derartigen Eintrages relativ einfach ist, wie Listing 3 zeigt.[2]

Abbildung 12: Pybliographic zum Verwalten einer Literaturdatenbank

Listing 3: Beispiel-Eintrag aus einer Literaturdatenbank

```
1 @Book{butcher-81,
2   title = "Copy-editing",
3   author = "Judith Butcher",
4   publisher = "Cambridge University Press",
5   year = "1981",
6   edition = "2nd",
7 }
```

Dazu existieren verschiedene Programme, die diese Verwaltung der Datenbank erleichtern. Mit LyX braucht dann nur aus dieser Datenbank der entsprechende Eintrag ausgewählt zu werden, der dann als Literaturzitat in den laufenden Text eingefügt wird. Es ist aber auch ein paralleles Arbeiten mit einem Literaturdatenbankprogramm wie *Pybliographer* (Abb. 12) möglich, welches das piping unterstützt und so direkt Zitate über den sogenannten LyX-Server in den Text einfügt (Abb. 13). Diese Eigenschaft weisen mehrere der gängigen BibTeX-Programme auf, z.B.

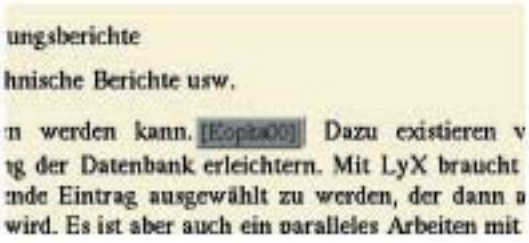


Abbildung 13: Einfügen von Literaturzitaten über das LyX-Piping

TkBibTEX[20] oder Sixpack[21]. Weitere Informationen gibt es unter *http://archiv.tu-chemnitz.de/pub/2001/0013/data/index.htm.*[22]

Will man auf eine externe Literaturdatenbank verzichten, so kann man mit dem direkt von LYX unterstützen Bibliography-Layout arbeiten, wo die Schlüssel sozusagen „online“ vergeben werden (Abb. 14).

Abbildung 14: Absatzlayout Bibliography

Diese Vorgehensweise ist aber ausdrücklich nicht zu empfehlen, denn derartige Literaturstellen können dann nur noch mit der „Drag and Drop“-Methode in andere Dokumente eingefügt werden. Dazu muss man aber in jedem Fall immer den Namen des alten Dokuments parat haben.

Rechtschreibüberprüfung

Da LYX keine eigene Rechtschreibkorrektur verwendet, kann der Spellchecker vom Benutzer frei vorgegeben werden und ist per Voreinstellung *ispell*, der allgemein sehr häufig zur Anwendung kommt, was aber eine rein subjektive Entscheidung ist. LYX unterstützt in jedem Fall den gewählten Spellchecker, wenn er sich über eine Kommandozeile starten lässt. Die neue Deutsche Rechtschreibung steht selbstverständlich zur Verfügung.
Abb. 15 zeigt die möglichen Optionen, die allgemein für jeden Spellchecker gültig sind.

Die Durchführung der Rechtschreibprüfung ist dann faktisch identisch zu dem sonst üblichen Verfahren und braucht hier nicht weiter erwähnt zu werden (Abb. 16).

Druckvorschau

Die Druckvorschau erfolgt völlig unabhängig vom LYX-System und damit auch unabhängig von LATEX bzw. TEX. Verwendet wird aufgrund der Tatsache, dass LATEX eine Datei im dvi-Format erzeugt, der *xdvi*-Viewer. Nachteilig ist hierbei, dass bei einem fehlerhaften LATEX-Lauf keine Druckvorschau möglich ist, so dass das Auffinden der Fehler etwas erschwert wird. Alternativ zur dvi-Ausgabe kann auch durch die von LYX durchgeführte Umwandlung ins Postscript-Format auch eine Vorschau mit Ghostview[23] erreicht werden, was insbesondere bei gedrehten Tabellen, Grafiken oder bei Farben zu empfehlen ist, da der dvi-Viewer nicht über entsprechende Fähigkeiten verfügt, um dies anzuzeigen.

Drucken

Abb. 17 zeigt, dass das Druckmenü im wesentlichen die Punkte enthält, wie sie allgemein üblich sind. Bei einer Ausgabe in eine Datei kann anschließend außerhalb von LYX mit einem externen Shellscript eine zweiseitige DIN-A5-Ausgabe in Buchform[24] erreicht werden. Steht kein Drucker mit Duplex-Einheit zur Verfügung, so kann mithilfe von *psnup* eine getrennte Ausgabe von Vor- und Rückseite erfolgen.

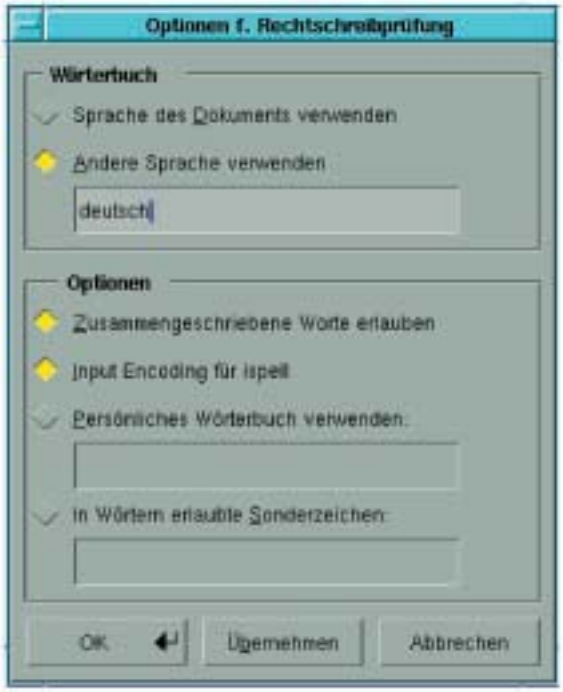


Abbildung 15: Allgemeine Optionen des Spellcheckers

Abbildung 16: Durchführung der Rechtschreibprüfung

Unabhängig von den nachträglichen Möglichkeiten mit Postscriptdateien, stellt LATEX mehrere Pakete bereit, die die globale Ausgabeform beeinflussen können, wie Verkleinerung auf DIN A5, Rahmen um jede Seite, usw.

Import - Export

LYX unterstützt hier verschiedene Verfahren direkt per Menüauswahl, wobei das zugrundeliegende externe Programm frei gewählt werden kann, was insbesondere für die Konvertierung nach HTML von Bedeutung ist, da die Verfahren sich erheblich unterscheiden.

---

20 http://www.cat.csiro.au/cmst/staff/pic/tkbibtex.html
21 http://www.santafe.edu/~dirk/sixpack/
22 Insbesondere das von der American Math Society herausgegebene Abkürzungsverzeichnis von Zeitschriften kann einem die Tipparbeit erheblich erleichtern.
Das Verzeichnis ist zum einen Teil einer jeden TEX-Distribution oder von http://www.ams.org/tex zu beziehen. Eine sehr gute Webseite ist auch http://liinwww.ira.uka.de/bibliography/.
23 http://wwwthep.physik.uni-mainz.de/~plass/gv/
24 Das erste Blatt enthält dann die ersten beiden und die letzten beiden Seiten, wobei die Gesamtzahl der Seiten auf ein Vielfaches von vier aufzurunden, bzw. durch leere Seiten zu ergänzen ist.

Listing 4: Shell Script zur Ausgabe einer PS-Datei in Buchform

```
1  #!/bin/sh
2  # script for book generation.
3  # Examples of use:
4  #
5  # man -t <command> | book4 | lpr -P<printer>
6  #     This will print the man pages in book format and
7  #     both sides. If you need file out, say:
8  # man -t <command> | book4 > out.ps
9  # cat name.ps | book4 > name2.ps # ps-file to book-format
10 # book4 name.ps | lpr            # same
11 #
12 PSBOOK=/usr/bin/psbook
13 PSNUP=/usr/bin/psnup
14 PSTOPS=/usr/bin/pstops
15
16 if test $# -gt 0; then
17   for N in $*; do
18 $PSBOOK -q $N | $PSNUP -q -2 | $PSTOPS -q "2:0,1U (1w,1h)"
19   done
20 else
21   $PSBOOK -q | $PSNUP -q -2 | $PSTOPS -q "2:0,1U(1w, 1h)"
22 fi
```

Abbildung 17: Das Druckmenü

Da LyX zwar ein eigenes Dateiformat besitzt, dieses aber von keiner anderen derzeit bekannten Anwendungen benutzt wird, erfolgen Konvertierungen von LyX in ein anderes Format grundsätzlich über das TEX-Dateiformat. Daher erscheint bei den bereits vordefinierten Format-Konvertierungen (Abb.18), die über das Menü Edit ➪ Einstellungen ➪ Formatkonvertierung zu erreichen sind, das Format LyX nicht auf der linken Seite der tabellarischen Zuordnung.

Sehr häufig wird die Konvertierung ins PDF-Format benötigt, die zum einen mit dem Programm *ps2pdf*, zum anderen aber auch mit *pslatex* erfolgen kann, die sich wesentlich unterscheiden, so dass beide Möglichkeiten unter LyX bestehen.[25] Eine Standardfrage ist immer wieder die Möglichkeit der Konvertierung von bzw. nach Microsofts Format für WinWord-Dateien. Es muss klar festgestellt werden, dass es keine befriedigende Lösung dafür gibt und im Extremfall, wenn komplizierte Formeln und umfangreiche Tabellen vorhanden sind, sowieso über das RTF-Format gegangen werden muss. Andererseits kann festgestellt werden, dass einfache Texte damit erfolgreich konvertiert werden können, jeweils in der Zuordnung RTF ⇔ TEX. Die anschließende Wandlung nach LyX kann über den sogenannten *ReLyX*-Import erfolgen. Direkte Unterstützung bietet LyX für Chess-Diagramme, Rastergrafiken und XFig-Dateien.[26] Insbesondere die Dokumentation von Schach-Partien wird dadurch sehr vereinfacht.

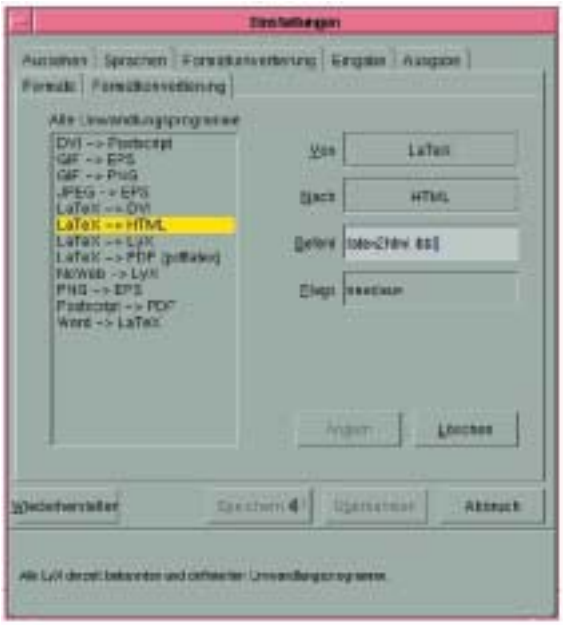


Abbildung 18: Unterstützte Import-Export-Möglichkeiten

Fonts - nicht ohne APL

Natürlich gibt es auch unter LATEX einen APL-konformen Zeichensatz, der durch Tab. 2 vollständig angegeben ist. Dabei existieren für diesen hier angegebenen Zeichensatz nur Großbuchstaben in Italic. *SAXPSA* ist ein weiterer APL-Zeichensatz, der auch die obere Hälfte des 255-Zeichen umfassenden Fonts belegt, so dass Klein- und Großbuchstaben existieren.[5] Allerdings sind dort alle weiteren Sonderzeichen mit längerem Namen als belegt, was zwar die Übersichtlichkeit erhöht, aber erheblich mehr Tipparbeit erfordert.

falsecollapsed false

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0		⎕	∇	∆	α	ω	∈	¨	∨	◊	≤	≥	ρ	∿	∘	○
1	∪	∩	⊂	⊃	_	¯	~	≠	⊤	⊥	⊣	⊢	⌊	⌈	→	↓
2		!	⍝	×	$	÷	∧	'	(	)	*	+	,	−	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
6		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}		

Tabelle 2: Der APL-Zeichensatz cmapl10

25 Weitere Hinweise gibt es bei: http://www.daszler.de/linux/pdf.html
26 diese werden mit *convert* in das eps-Format konvertiert

Der entsprechende Zeichensatz ist *cmapl10* und kann wie jeder andere Zeichensatz auf jede beliebige Größe expandiert werden, solange entsprechende Informationen für Metafont zur Verfügung stehen. ⍝ wäre dann z.B. ein 24pt hohes Kommentarzeichen aus dem APL-Zeichensatz. Eine Ausgabe des Quellcodes für Veröffentlichungen lässt sich dann ebenfalls realisieren:

```
R←GETDATA file;H;I;LB;SV;T;U;del;label

⍝Get Data-frame from IBM-211-objekt-file
⍝Version economically disk memory (SM)
 del←211 SVOFFER 'SV'
 →(del≠2)/ER1
 I←0 ⋄ R←''                    ⍝Initiate
 SV←'USE' file ⋄→(0≠↑SV)/ER2 ⍝Return code
 SV←'LIST' 'NAMES' ⋄←SV        ⍝List library
 LB←(H≠' ')CR←,H,''            ⍝Set labels
S:SV←'GET'((I←I+1)⊃LB)⋄T←2⊃SV ⍝Get data
 →(0=0T)/NUM                   ⍝Numeric?
 U←1⊃T ⋄u←(U≠' ')CU ⋄→OK      ⍝Partition
NUM:U←1⊃T                      ⍝Numeric variable
OK:R←R,CU[(4⊃T)⍴⍬(3⊃T)?2⊃T]  ⍝Puzzle
 →(I<⍴LB)S                     ⍝Nr. of columns
 R←LB,[1]⊃[1]R ⋄→0             ⍝Form data
frame
ER1:'not shared' ⋄→0
ER1:'Pfad oder Datei nicht gefunden' ⋄→0
```

Sämtliche Sonderzeichen können über einen Zweizeichencode eingebunden werden. Eine entsprechende Zusammenstellung zeigt Tabelle 3. Weiterhin stehen alle Buchstaben in der Version mit Unterstrich zur Verfügung. Wie

APL	Code	Bezeichnung	APL	Code	Bezeichnung
⍴	\RO	rho	∪	\UU	up U
⍳	\IO	iota	⊢	\LK	left tack
⎕	\BX	quad box	⊣	\RK	right tack
⌈	\CE	ceiling	_	\US	underscore
⌊	\FL	floor	¯	\NG	high minus
⊥	\DE	decode	¨	\DI	dieresis
⊤	\EN	encode	⍺	\AM	alpha
∇	\DL	del	⍵	\OM	omega
∆	\LD	delta	∘	\SO	small circle
~	\NT	not	⍉	\TR	transpose
○	\LO	circle	⌽	\RV	reverse
→	\GO	arrow right	⊖	\CR	col reverse
∨	\OR	logical or	⍋	\GD	grade down
⋄	\DM	diamond	⍒	\GU	grade up
≤	\LE	less equal	⍕	\FM	format
≥	\GE	greater equal	⍎	\XQ	execute
|	\AB	stile	⊂	\SS	subset
{	\LB	left brace	⊃	\CO	contains
}	\RB	right brace	\	\CB	col backslash
↓	\DA	arrow down	/	\CS	col slash
↑	\UA	arrow up	⌶	\IB	I-beam
∊	\EP	epsilon	⌹	\DQ	divide quad
≠	\NE	not equal	⎕	\QQ	quote quad
\	\BL	backslash	⍞	\PD	protected del
⊂	\RU	right U	⍱	\NR	nor
⊃	\LU	left U	⍲	\NN	nand
∩	\DU	down U	⍟	\LG	logarithm

Tabelle 3: Zweizeichencode der APL-Symbole

für alle TEX-Zeichensätze lassen sich auch mit dem APL-Zeichensatz beliebige Fontgrößen erzeugen (Tabelle 4).

Die LATEX - Präambel

Diese stellt das eigentliche Interface zwischen LYX und LATEX dar, denn hier müssen alle TEX/LATEX-spezifischen Definitionen angegeben werden. Ein großer Teil dieser Angaben könnte auch im TEX-Modus im LYX-Text erscheinen, was jedoch nicht zu empfehlen ist, da dadurch die Übersicht nicht unbedingt erhöht wird. Die Latex Präambel ist integraler Bestandteil einer jeden TEX-Datei, kann aber theoretisch trotzdem leer bleiben, wenn mit der Festlegung der Dokumentenklasse bereits alles festliegt. Dies gilt nicht für die LYX-Dokumente, da hier mindestens der Befehl für die Schreibweise von LYX aufgenommen wird, damit es nicht LYX geschrieben wird.

Abbildung 19: Teil des Inhalts der LATEX - Präamble dieses Textes

Grundsätzlich ist die Präambel (LATEX-Vorspann) nichts weiter als ein einfaches Editor-Fenster indem insbesondere die benötigten Zusatzpakete anzugeben sind, jedoch nur die, die LYX nicht von selbst berücksichtigt. Diese einfachen Editierfunktionen sind sicherlich keine Stärke von LYX, man lebt mit ihnen. Grundsätzlich kann per `\input`-Befehl die gesamte Präambel als externe Datei eingelesen werden und somit auch durch andere externe Editoren bearbeiten werden. Texte mit komplexem Layout haben auch in der Regel ein komplexe Präambel zur Folge.

Fehleranalyse

Bei einem sogenannten LATEX-Lauf betreibt LYX das LATEX-Makro, welches wiederum den TEX-Lauf organisiert, im sogenannten Batch-Modus. Die dabei auftretenden Feh-

Command	\aplXXX\apl labû\qor /(^\fnt\qeps lab)^fnt=':'
\aplTiny	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplScript	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplFootnote	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplSmall	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplNormal	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplLarge	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplLARGE	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplHuge	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplHUGE	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplMonster	lab←∨/(∧\fnt∊lab)∧fnt=':'
\aplMONSTER	lab←∨/(∧\fnt...

Tabelle 4: Verschiedene Fontgrößen mit dem APL-Stylefileapl2.sty (http://www.perce.de/lyx/apl2.sty.gz)

ler versucht LyX möglichst genau zu lokalisieren und im Dokument an entsprechender Stelle zu markieren. Wird zum Beispiel an dieser Stelle des LyX-Textes der nicht-existierende LaTeX-Befehl `\uhu` angegeben, so sollte im Idealfall die Fehlermarkierung auch an diese Stelle gesetzt werden. Tatsächlich wird sie jedoch, wie Abb. 20 zeigt, an anderer Stelle plaziert. Das zweimalige Auftreten der Sequenz `\uhu` darf hier nicht irritieren, die erste ist reiner Text und die zweite ein im späteren Text unsichtbarer LaTeX-Befehl und daher in LyX grundsätzlich in roter Farbe.[27] Klickt man mit der Maus auf die Fehlermarkierung erhält man jedoch eine relativ genaue Fehlerbeschreibung (Abb. 21), so dass eine eingehende und schnelle Analyse mit anschließender Fehlerbeseitigung möglich ist.

Ein LaTeX-Lauf entspricht prinzipiell einem Übersetzungsvorgang wie man ihn hinreichend von Programmsprachen kennt, so dass hier auch mit sehr vielen Folgefehlern gerechnet werden kann, die bei längeren Texten leicht die Zahl von 100 und mehr erreichen können. Dies ist grundsätzlich kein Problem, da die Fehler in ihrer Reihenfolge *einzeln* abgearbeitet werden, so dass sich Folgefehler bei erneuten Übersetzungsversuchen von selbst erledigen.

Abbildung 20: Fehlerbehandlung durch LyX

Abbildung 21: Fehlermeldung

Navigation

Das Navigieren innerhalb umfangreicher Texte ist manchmal nicht einfach, wird daher von LyX durch einen eige-

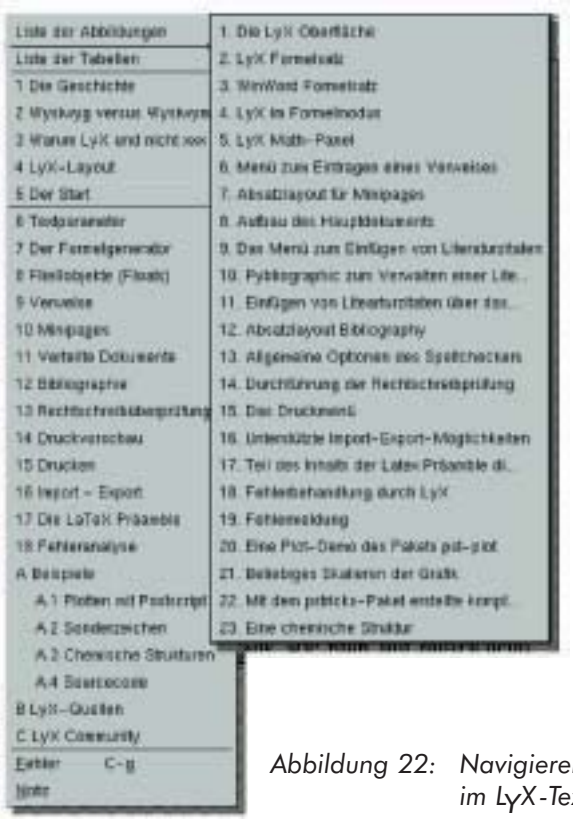


Abbildung 22: Navigieren im LyX-Text

27 Vgl. entsprechenden Abschnitt auf Seite 40 zum ERT.

nen Menüpunkt unterstützt. Neben sämtlichen Kapitel- und Unterkapitelüberschriften werden sowohl alle Tabellen als Abbildungen aufgeführt, so dass man faktisch an jeden Punkt des Textes springen kann.

Gleiches lässt sich auf ähnliche Weise mit dem Menüpunkt „Bearbeiten & Suchen und Ersetzen“ erreichen, worauf hier nicht weiter eingegangen zu werden braucht, da es zu den Standardoptionen einer jeden Textverarbeitung gehört. In der jetzigen Version unterstützt LyX leider noch kein Ersetzen von Kommandosequenzen, sondern nur von Text, der auch auf dem Bildschirm zu sehen ist.

Dokumentation

Ein großes Problem bei Open Source Projekten ist immer die Dokumentation und Hilfe, die über Mailinglisten hinausgeht. Zum einen kommen die Freiwilligen, die sich um die Dokumentation kümmern, meistens nicht hinter den Entwicklern mit ihrem dynamischen Einwicklungsprozess hinterher und zum anderen lesen die wenigsten die Dokumentation, sondern gehen gleich mit einer Frage an die Mailingliste, die dort dann zum 1000. Mal gestellt wird. LyX hat über den Menüpunkt Hilfe eine Auswahl an Dokumenten bzw. Hilfedateien, die selbst als LyX-Dateien vorliegen, also jederzeit ausgedruckt werden können. Der Umfang ist beträchtlich, leidet jedoch unter einer leicht chaotischen Strukturierung, weshalb der Anwender häufig Probleme hat, die richtigen Antworten zu seinen Fragen zu finden.

Abbildung 23: Hilfedateien und LyX

Da die derzeit aktuelle Version 1.1.6 erhebliche Unterschiede zur Vorgängerversion aufweist, besteht in einzelnen Punkten der Dokumentation noch Verbesserungsbedarf, der im Moment zeitlich aufgeschoben wird, da die neue 1.2 Version wiederum einen kleinen Quantensprung darstellen soll.

Anhang

Beispiele

Nachdem im extremen Schnelldurchgang die grundlegenden Eigenschaften von LyX beschrieben wurden, sollen anhand einiger willkürlich ausgewählter Beispiele, die Vorzüge eines Satzsystems wie TEX im Zusammenhang mit einem Dokumentenprozessor wie LyX verdeutlicht werden. Diese Beispiele haben natürlich nur sekundär etwas mit LyX zu tun, denn mit einem anderen Texteditor, z.B. emacs ist dasselbe Ergebnis möglich, wenn auch bei einem ganz anderen Umfeld.

✦ Plotten mit Postscript

Die folgenden Plots sind mit dem LATEX-Paket *pst-plot.sty*[28], welches selbst Teil des umfangreichen Pakets *pstricks.sty*[29] ist. Einige Pakete werden direkt von LyX unterstützt, andere können über die sogenannte LATEX-Präambel geladen werden, während der reine LATEX-Code (**E**VAL **R**ED **T**EXT) direkt in das LyX-Dokument eingebunden wird.

Dieser erscheint dann im LyX-Layout als roter Text, weshalb auch von ERT gesprochen wird. Die Abbildung 24 ist eine derartige Anwendung. Anstelle erst mit einem Vektorzeichenprogramm wie XFig oder anderen eine entsprechende Grafik zu erstellen, wird diese gleich mithilfe der vom Umfang her mächtigen Postscript-Befehle erstellt. In der sogenannten LATEX-Präambel muss dem LATEX-System bekannt gegeben werden, welche externen Pakete hinzugeladen werden sollen:

```
\usepackage{pst-plot}
\usepackage{pstricks}
```

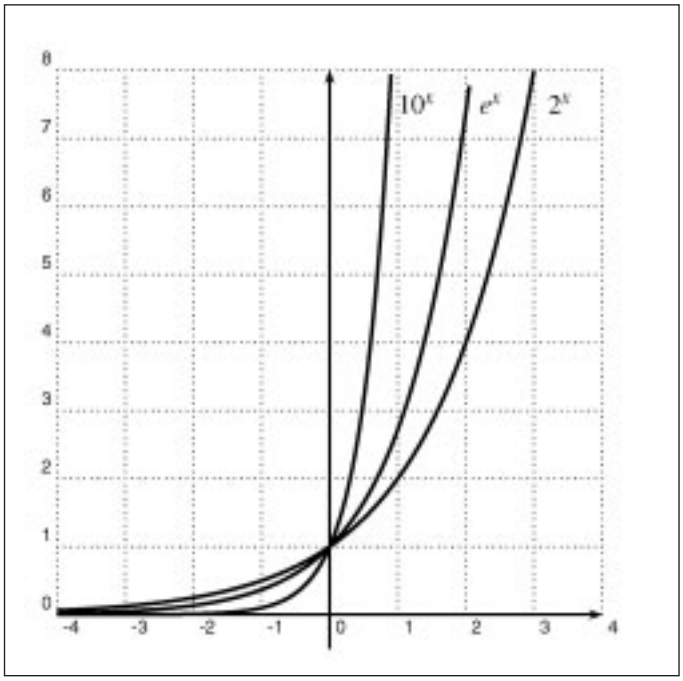


Abbildung 24: Eine Plot-Demo des Pakets pst-plot

28 ftp://ftp.dante.de/tex-archive/graphics/pstricks/latex/pst-plot.sty
29 ftp://ftp.dante.de/tex-archive/graphics/pstricks/

Listing 5: Postscript Befehle anwenden

```
1 \begin{center}
2 \begin{pspicture}(-4,0)(4,8)%                    KO-System
3 \psset{xunit=0.9cm,yunit=0.9cm}%                 Skalieren
4 \psgrid[subgriddiv=1,griddots=10,gridlabels=7pt](-4,-0.5)(4,8)
5 \psline[linewidth=1pt]{->}(-4,0)(+4,0)
6 \psline[linewidth=1pt]{->}(0,-0.5)(0,8)
7 \psplot[plotstyle=curve,linewidth=1.5pt]{-4}{0.9}{10 x exp}%
      postscript function
8 \rput[l](1,7.5){$10^x$}
9 \psplot[plotstyle=curve,linewidth=1.5pt]{-4}{3}{2 x exp}%
      postscript function
10 \rput[l](2.2,7.5){$e^x$}
11 \psplot[plotstyle=curve,linewidth=1.5pt]{-4}{2.05}{2.7183 x exp}%
      postscript function
12 \rput[l](3.2,7.5){$2^x$}
13 \end{pspicture}
14 \end{center}
```

Listing 6: Befehlssequenzen für Abb. 26

```
1\begin{pspicture}(0,0)(6.5,-9)
2\psframe[linewidth=1pt](0,0)(6.4,-0.5)
3\psline(1.6,0)(1.6,-0.5)
4\psline(3.2,0)(3.2,-0.5)
5\psline(4.8,0)(4.8,-0.5)
6\rput(0.8,-0.25){$X_0$}\rput(2.4,-0.25){$X_1$}
7\rput(4,-0.25){$X_2$}\rput(5.6,-0.25){$X_3$}
8\rput(3.2,0.25){Klartextblock: 4*16 Bit}
9\psline{->}(0.8,-0.5)(0.8,-0.85)\rput(0.8,-1){$\bigodot$}
10\psline{->}(-0.5,-1)(0.65,-1)\rput(-0.75,-1){$K_0$}
11\psline{->}(0.8,-1.15)(0.8,-6.35)\myExOr{0.8}{-6.5}
12\psline{->}(0.8,-6.65)(0.8,-8)
13\psline{->}(2.4,-0.5)(2.4,-1.35)\rput(2.4,-1.5){$\bigoplus$}
14\psline{->}(-0.5,-1.5)(2.25,-1.5)\rput(-0.75,-1.5){$K_1$}
15\psline{->}(2.4,-1.6)(2.4,-5.35)\myExOr{2.4}{-5.5}
16\psline{->}(2.4,-5.65)(2.4,-7)(4,-7.5)(4,-8)
17\psline{->}(5.6,-0.5)(5.6,-0.85)\rput(5.6,-1){$\bigodot$}
18\psline{->}(6.9,-1)(5.75,-1)\rput(7.15,-1){$K_3$}
19\psline{->}(5.6,-1.15)(5.6,-6.35)\myExOr{5.6}{-6.5}
20\psline{->}(5.6,-6.65)(5.6,-8)
21\psline{->}(4,-0.5)(4,-1.35)\rput(4,-1.5){$\bigoplus$}
22\psline{->}(6.9,-1.5)(4.15,-1.5)\rput(7.15,-1.5){$K_2$}
23\psline{->}(4,-1.6)(4,-5.35)\myExOr{4}{-5.5}
24\psline{->}(4,-5.65)(4,-7)(2.4,-7.5)(2.4,-8)
25\pscircle(0.8,-2.5){0.05}\pscircle(4,-2){0.05}
26\psline{->}(0.8,-2.5)(1.45,-2.5)\myExOr{1.6}{-2.5}
27\psline{->}(4,-2)(3.5,-2)(2.9,-2.5)(1.75,-2.5)
28\pscircle(5.6,-2.5){0.05}\pscircle(2.4,-2){0.05}
29\psline{->}(2.4,-2)(2.9,-2)(3.5,-2.5)(4.65,-2.5)
30\psline{->}(5.6,-2.5)(4.95,-2.5)\myExOr{4.8}{-2.5}
31\psline{->}(1.6,-2.65)(1.6,-3.35)\rput(1.6,-3.5){$\bigodot$}
32\psline{->}(-0.5,-3.5)(1.45,-3.5)\rput(-0.75,-3.5){$K_4$}
33\psline{->}(1.6,-3.65)(1.6,-4.35)\rput(1.6,-4.5){$\bigoplus$}
34\psline{->}(1.6,-4.65)(1.6,-5.5)(2.25,-5.5)
35%
36\pscircle(1.6,-5.5){0.05}
37\psline{->}(1.6,-5.5)(1.6,-6)(2.9,-6)(3.5,-6.5)(5.45,-6.5)
38\psline{->}(4.8,-2.65)(4.8,-3.35)\rput(4.8,-3.5){$\bigoplus$}
39\pscircle(1.6,-4){0.05}
40\psline{->}(1.6,-4)(2.4,-4)(2.9,-4)(3.5,-3.5)(4.65,-3.5)
41\psline{->}(6.9,-4.5)(4.95,-4.5)\rput(7.15,-4.5){$K_5$}
42\psline{->}(4.8,-3.65)(4.8,-4.35)\rput(4.8,-4.5){$\bigodot$}
43\psline{->}(4.8,-4.65)(4.8,-5.5)(4.15,-5.5)
44\pscircle(4.8,-5){0.05}
45\psline{->}(4.8,-5)(3.5,-5)(2.9,-4.5)(1.75,-4.5)
46\pscircle(4.8,-5.5){0.05}
47\psline{->}(4.8,-5.5)(4.8,-6)(3.5,-6)(2.9,-6.5)(0.95,-6.5)
48\psframe[linewidth=1pt](0,-8)(6.4,-8.5)
49\psline(1.6,-8)(1.6,-8.5)\psline(3.2,-8)(3.2,-8.5)
50\psline(4.8,-8)(4.8,-8.5)
51\rput(0.8,-8.25){$C_0^0$}\rput(2.4,-8.25){$C_1^0$}
52\rput(4,-8.25){$C_2^0$}\rput(5.6,-8.25){$C_3^0$}
53\rput(3.2,-9){Chiffreblock: 4*16 Bit}
54% die einzelnen Ebenen markieren
55\rput(3.2,-1){\footnotesize 1}
56\rput(3.2,-1.5){\footnotesize 2}
57\rput(3.2,-2.5){\footnotesize 3}
58\rput(3.2,-3.5){\footnotesize 4}
59\rput(3.2,-4.5){\footnotesize 5}
60\rput(3.2,-5.5){\footnotesize 6}
61\rput(3.2,-6.5){\footnotesize 7}
62\end{pspicture}
```

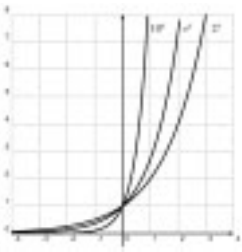


Abbildung 25: Beliebiges Skalieren der Grafik

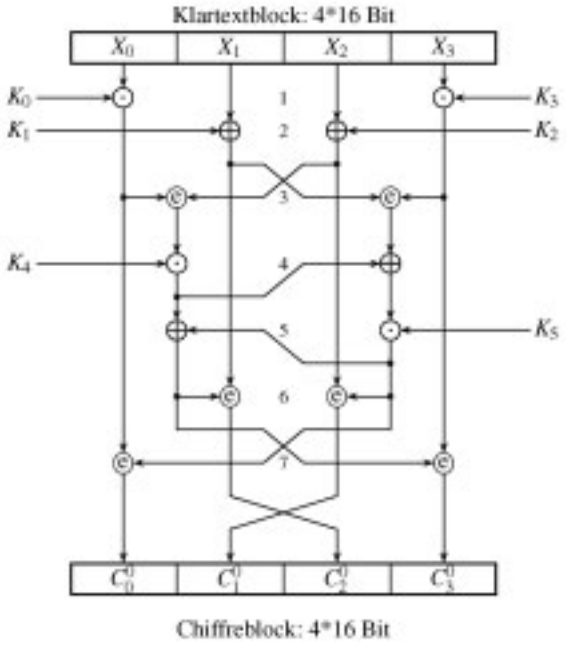


Abbildung 26: Mit dem pstricks-Paket erstellte komplexe Grafik (Formaler Aufbau des IDEA-Verschlüsselungsalgorithmus)

Aufgrund der verwendeten Vektorgrafik kann das Bild beliebig verkleinert werden, was über einen Befehl möglich ist:

```
\psset{xunit=0.25cm,yunit=0.25cm}.
```

Der LaTeX-Code (vgl. Listing 5) ist relativ einfach, wenn man berücksichtigt, dass für die mathematischen Operationen von Postscript die **U**MKEHRTE **P**OLNISCHE **N**OTATION benutzt wird, was für Besitzer von HP-Taschenrechnern kein Problem darstellen dürfte.

Komplexe Liniengrafiken erfordern in der Regel ebenso komplexe Befehlsfolgen was durch Abb. 26 bestätigt wird. Verwendet wird unter anderem auch das selbst definierte Exklusiv-Oder-Symbol ⊕, welches über `\exor` aufgerufen werden kann.

✦ Sonderzeichen

Häufig benötigt man eigene Sonderzeichen, die dann meistens über eine kleine gif-Datei eingebunden werden. Unter TeX ist dies in der Regel kein großes Problem, denn man kann sein eigenes in keiner Fontdatei vorhandene Zeichen einfach selbst definieren. Exemplarisch soll dies für das Pentagon-Symbol gezeigt werden, in das aber die aktuelle Sei-

tennummer eingetragen werden soll. Nur scheinbar ein größeres Problem, denn das Pentagon ist als Fünfeck völlig symmetrisch und der Radius des Außenkreises braucht nur der Breite oder der Höhe der Seitenzahl angepasst zu werden.

In der LaTeX-Präambel wird das neue Zeichen `\myPentagonPage` definiert (Listing 7). Als Seitenzahlen würden dann die folgenden Symbole erscheinen:

Listing 7: Befehlssequenz für das Pentagonzeichen mit Seitennummer

```
 1 \usepackage{ifthen}
 2 \newlength{\myL}\newlength{\myLL}
 3 \newcommand\myPentagonPage{%
 4 \settowidth{\myL}{\thepage}%  Breite
 5 \settoheight{\myLL}{\thepage}%  Hoehe
 6 \ifthenelse{\lengthtest{\myL>\myLL}}{\setlength{\myL}
   {0.8\myL}}  {\setlength{\myL}{0.8\myLL}}
 7 \psset{unit=1.0}
 8 \begin{pspicture}(-0.9511\myL,-0.8090\myL)(0.9511
   \myL,\myL)
 9 \psline(0.5877\myL,-0.8090\myL)(-0.5877\myL,-0.8090
   \myL)%
10 (-0.9511\myL,0.3090\myL)(0,\myL)%
11 (0.9511\myL,0.3090\myL)(0.5877\myL,-0.8090\myL)
12 \rput(0,0){\thepage}
13 \end{pspicture}%
14 }
```

✦ Chemische Strukturen

Hierfür gibt es diverse Zusatzpakete, wie *chemtex*, *xymtex*, usw., die alle über CTAN[30] öffentlich zur Verfügung stehen. Die Anweisung `\usepackage{epic,carom,hetarom}` in der LATEX-Präambel und die folgende einfache LaTeX-Befehlssequenz

`\decaheterov[aegi]{4==0}{3D==0}`

führt zur chemischen Struktur in Abbildung 27.

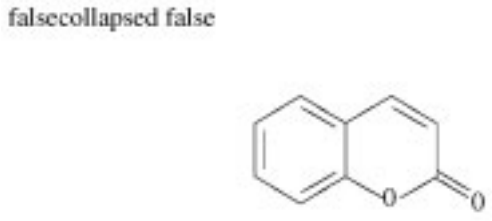


Abbildung 27: Eine chemische Struktur

✦ Sourcecode

Einer der häufigen Anwendungsfälle ist das Zitieren der Quellcodes der verschiedenen Programmiersprachen, wobei man insbesondere einen Teil der Syntax optisch durch Fett- oder Kursivschrift hervorheben will. Das LaTeX-Paket *listings* unterstützt eine Vielzahl von Programmiersprachen für dieses „Highliting“ und bietet daneben noch sehr viel mehr Möglichkeiten für eine gute Ausgabe von Quellcode. Zur Zeit wereden folgende Sprachen mit ihren jeweiligen Dialekten unterstützt:

Algol (60, 68)
C (ANSI, Objective)
C++ (ANSI, Visual)
Caml (light, Objective)
Clean
Cobol (1974, 1985, ibm)
Comal 80
csh
Delphi
Eifel
Elan
Euphoria
Fortran (77, 90, 95)
Haskell
HTML
IDL (empty, CORBA)
Java
Lisp
Logo
make (empty, gnu)
Mathematica (1.0, 3.0)
Matlab
Mercury
Miranda
ML
Modula-2
Oberon-2
OCL (decorative, OMG)
Pascal (Borland6, Standard, XSC)
Perl
PL/I
POV
Prolog
Python
SHELXL
Simula (67, CII, DEC, IBM)
SQL
TEX (AlLATEX, common, LATEX, plain, primitive)
VBScript
VHDL

Exemplarisch ist hier durch Listing 8 ein kleines Pythonprogramm angegeben, welches den Faxsupport für LyX-Dateien unterstützt. Selbstverständlich lässt sich ein Source-

Listing 8: Faxsupport für LyX ab Version 1.1.6

```
 1  #!/usr/bin/python
 2  from Tkinter import *
 3  import os,sys
 4
 5  class MyDialog:
 6      def __init__(self, parent):
 7          Label(parent, text="Enter phone number").pack()
 8
 9          self.e = Entry(parent)
10          self.e.pack(padx=5)
11          self.e.focus_set()
12
13          Button(parent, text="Send", command=self.send).pack
                (pady=5)
14
15      def send(self):
16          phone = self.e.get()
17          if len(phone) == 0:
18              return
19          os.system("faxspool " + phone + " " + sys.argv[1])
20          sys.exit(0)
21
22  root = Tk()
23  MyDialog(root)
24  mainloop()
```

30 COMPREHENSIVE TEX ARCHIVE NETWORK http://www.ctan.org
31 Z.B. bei http://www.lob.de. Das sogeannnte TeX-Livesystem ermöglicht für verschiedene Betriebsysteme das Arbeiten direkt von der CD.

codelisting auch für APL erreichen, bislang hat sich nur noch keiner Gedanken darüber gemacht, welche Schlüsselwörter hervorgehoben werden sollten.

✦ LYX-Quellen

Unter *http://www.lyx.org* erhält man alle wesentlichen Informationen, die man für die Installation von LyX benötigt. Installationen gibt es prinzipiell für alle gängigen Betriebssysteme, einschließlich OS/2 und Windows, wobei hier allerdings ein X-Server benötigt wird. Vorausgesetzt wird grundsätzlich ein vollständig installiertes TEX-System, welches kostenlos von *http://www.dante.de* per ftp heruntergeladen werden kann oder auch auf CD zu erwerben ist.[31] LyX kann für Standardinstallationen von Linux auch als übersetzte Binärdatei per ftp von *ftp://ftp.sylvan.com* heruntergeladen werden.

Unter *http://www.educat.hu-berlin.de/~voss/lyx/download.html* sind weitere Downloadquellen, insbesondere für spezielle Versionen von LyX, angegeben.

Weiterhin werden verschiedene Hilfsprogramme *xdvi* und *ghostscript* benötigt, ohne die allerdings auch kein TEX-System auskommt. Hilfreiche Programme sind auch die Postscript Utilities, wie *psnups*, *psresize*, usw.

✦ LYX Community

LyX als Open-Source-Projekt ist auf die intensive Mitarbeit der Nutzer in der Kommunikation bei Anwendung und Entwicklung angewiesen. Die Homepage *http://www.lyx.org* wurde bereits erwähnt. Daneben gibt es insbesondere Mailinglisten für:

- *Anwender:*
  lyx-users@lists.lyx.org/ (High Volume)
- *Entwickler:*
  lyx-devel@lists.lyx.org/ (Medium/High Volume)
- *Dok-Team:*
  lyx-doc@lists.lyx.org/ (Low Volume)

Das Eintragen in diese Mailinglisten geschieht wie allgemein üblich, kann aber auch über die LyX-Homepage erfolgen. Eine umfangreicheWebsite mit Tips und Tricks zu LyX/LATEX findet man unter *http://www.educat.hu-berlin.de/~voss/lyx/*. Zu allen Listen gibt es auch Online-Archive, die unter *http://www.mail-archive.com/lyx-XXX@lists.lyx.org/* errreicht werden können, wobei „*XXX*“ durch „*devel*“, „*user*“ oder „*doc*“ zu ersetzen ist.

Mittelfristiges Ziel der LyX-Entwicklung ist die Fähigkeit in jeder beliebigen GUI-Umgebung (GRAPHICAL USER INTERFACE) zu laufen. Ein ständiges Thema ist selbstverständlich die Integration von LATEX-Paketen, so dass dem Anwender auch hier mehr und mehr an Arbeit bzw. eingehenden LATEX-Kenntnissen erspart bleiben soll.

Literatur

[1] Ingo Klöckl: LATEX *2e: Tips und Tricks*. Heidelberg. dpunkt Verlag, 2000.

[2] Helmut Kopka: *LaTeX Band 1 - Einführung*. Addison-Wesley, München, 3. edition, 2000.

[3] Leslie Lamport: *Das Latex Handbuch*. Addison-Wesley, München, 1995.

[4] Goossens, Michael; Mittelbach, Frank; Samarin, Alexander: *The LaTeX Companion*. Addison Wesley, 1994.

[5] Voß, Herbert: APL-Font SAXPSA (als pdf-Datei). http://www.perce.de /lyx/apl2.pdf, 2001.

Kontakt

Herbert Voß, Berlin
eMail:voss@lyx.org

Dieter Kilsch

Durchdringung von Kugel und Zylinder

Mit einem Programm die Masse berechnen

In diesem Beitrag wird die Masse berechnet, die von einem Kugelschleifer beim Schleifen eines Zylinder herausgeschliffen wird. Bei diesem Vorgang bewegt sich der Schleifkopf radial in den Zylinder hinein. Die Berechnung unterliegt der Voraussetzung, dass die Schleiftiefe kleiner als der Radius der Schleifkugel ist. Daraus folgt, dass alle geometrischen Parameter des Schleifvorgangs vom *tangentialen Durchmesser* des Bohrlochs abhängen. Dies ist der Durchmesser des Bohrlochs in derjenigen Ebene orthogonal zur Zylinderachse, die den Kugelmittelpunkt enthält. Es ist der größte Durchmesser des Bohrlochs in einer zur Zylinderachse orthogonalen Ebene. Der *axiale Durchmesser* des Bohrlochs ist parallel zur Zylinderachse gemessene Durchmesser. Seine Abhängigkeit vom tangentialen Durchmesser wird berechnet. Bekannt sind natürlich auch die Radien der Kugel und des Zylinders.

Festlegungen für die Berechnung

Zur Vereinfachung der Berechnung wird als wesentlicher Parameter der Halbmesser benutzt, das heißt der halbe Durchmesser.

Zur Berechnung wird ein Koordinatensystem eingerichtet, dessen z-Achse gleich der Symmetrieachse des Zylinders ist. Der Mittelpunkt der Bohrkugel liegt auf der positiven x-Achse.

Die Bohrrichtung ist also gleich der negativen x-Richtung. Die y-Achse vervollständigt das Koordinatensystem zu einem orthogonalen Rechtssystem.

Die Bezeichnungen werden gemäß Tabelle 1 festgelegt. Die erwähnte *Eindringtiefe* wird im xy-Schnitt ab der Schnittlinie der beiden Schnittkreise gemessen.

Variable	Bedeutung
α	tangentialer Halbmesser des Bohrlochs (gemessen)
A	axialer Halbmesser des Bohrlochs
r_K	Radius der Kugel (bekannt)
r_Z	Radius des Zylinders (bekannt)
e_K	Eindringtiefe in die Kugel
e_K	Eindringtiefe in den Zylinder
x_{MK}	x-Koordinate des Kugelmittelpunkts

Tabelle 1: Symbolverzeichnis

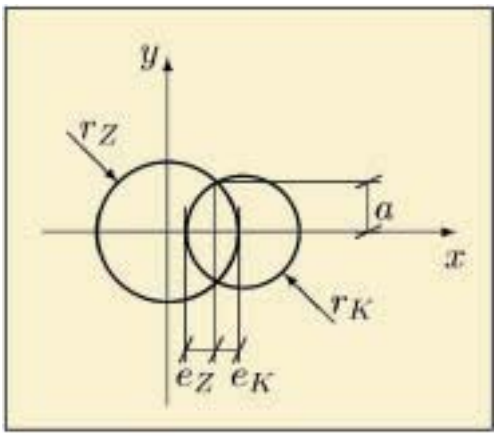


Abbildung 1: xy-Schnitt

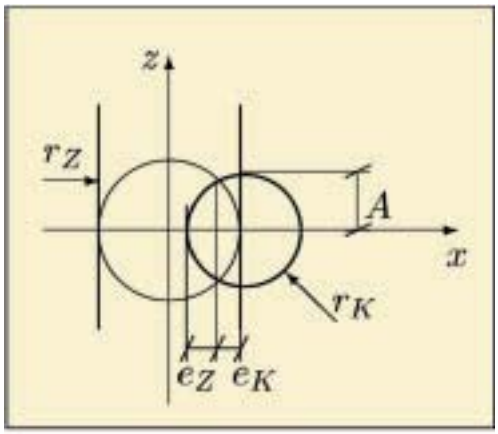


Abbildung 2: xz-Schnitt

Berechnung

Aus der Zeichnung ergibt sich:

$x_{MK} = r_Z - e_Z + r_K - e_K$

Mit Hilfe des Satzes von Pythagoras erkennt man in den jeweiligen Kreisen:

$$r_K^2 = (r_K - e_Z)^2 + a^2 \quad (2)$$
$$r_Z^2 = (r_Z - e_K)^2 + a^2 \quad (3)$$
$$r_K^2 = (r_K - e_Z - e_K)^2 + A^2 \quad (4)$$

Neben den offensichtlichen Bedingungen für diese Berechnung ($r_K \leq e_Z$ und $r_Z \geq e_K$), liefert die letzte Gleichung die stärkere Einschränkung für diese Berechnung:
$r_K \geq e_Z + e_K$

(2) und (3) werden umgeformt in die Gleichung des Kugelmittelpunkts (1) eingesetzt:

$$x_{MK} = r_z - e_K + r_K - e_Z = \sqrt{r_Z^2 - a^2} + \sqrt{r_K^2 - a^2}, \quad (5)$$

Aus den Gleichungen (2) - (4) folgen durch Umstellen:

$$e_Z = r_K - \sqrt{r_K^2 - a^2} \quad (6)$$
$$e_K = r_Z - \sqrt{r_Z^2 - a^2} \quad (7)$$
$$\begin{aligned} A^2 &= r_K^2 - (r_K - e_Z - e_K)^2 \\ &= r_K^2 - \left(r_K - r_K + \sqrt{r_K^2 - a^2} - r_Z + \sqrt{r_Z^2 - a^2}\right)^2 \\ &= r_K^2 - \left(\sqrt{r_K^2 - a^2} + \sqrt{r_Z^2 - a^2} - r_Z\right)^2 \\ &= r_K^2 - \left(r_K^2 - a^2 + r_Z^2 - a^2 + r_Z^2 + 2\sqrt{r_K^2 - a^2}\sqrt{r_Z^2 - a^2}\right. \\ &\qquad \left. -2r_Z\left(\sqrt{r_K^2 - a^2} + \sqrt{r_Z^2 - a^2}\right)\right) \\ &= 2a^2 - 2r_Z^2 - 2\sqrt{r_K^2 - a^2}\sqrt{r_Z^2 - a^2} + 2r_Z\left(\sqrt{r_K^2 - a^2} + \sqrt{r_Z^2 - a^2}\right) \end{aligned}$$
$$A = \sqrt{2}\sqrt{a^2 - r_Z^2 - \sqrt{r_K^2 - a^2}\sqrt{r_Z^2 - a^2} + r_Z\left(\sqrt{r_K^2 - a^2} + \sqrt{r_Z^2 - a^2}\right)} \quad (8)$$

Der Körper, dessen Volumen ermittelt werden soll, ist symmetrisch zur Aquatorialebene der Kugel (xy-Ebene) und zur Ebene durch die Zylinderachse und den Kugelmittelpunkt (*xz*-Ebene). Zur Volumenberechnung ermitteln wir daher lediglich den Teil mit nicht-negativen y- und z-Koordinaten und multiplizieren das Ergebnis mit 4. Die Stammfunktionen werden in Bronstein/Semendajew[1] nachgeschlagen:

$$\begin{aligned} V &= 4\int_0^a \int_{x_{MK}-\sqrt{r_Z^2-y^2}}^{\sqrt{r_Z^2-y^2}} \int_0^{\sqrt{r_K^2-(x-x_{MK})^2-y^2}} \mathrm{d}z\,\mathrm{d}x\,\mathrm{d}y \\ &= 4\int_0^a \int_{x_{MK}-\sqrt{r_Z^2-y^2}}^{\sqrt{r_Z^2-y^2}} \sqrt{r_K^2-(x-x_{MK})^2-y^2}\,\mathrm{d}x\,\mathrm{d}y \\ &= 4\int_0^a \int_{-\sqrt{r_K^2-y^2}}^{\sqrt{r_Z^2-y^2}-x_{MK}} \sqrt{r_K^2-u^2-y^2}\,\mathrm{d}u\,\mathrm{d}y \qquad (u = x - x_{MK}) \\ &= 4\frac{1}{2}\int_0^a \left[u\sqrt{r_K^2-u^2-y^2} + (r_K^2-y^2)\arcsin\left(\frac{u}{\sqrt{r_K^2-y^2}}\right)\right]_{u=-\sqrt{r_K^2-y^2}}^{u=\sqrt{r_Z^2-y^2}-x_{MK}} \mathrm{d}y \\ &= 2\int_0^a \left(\sqrt{r_Z^2-y^2} - x_{MK}\right)\sqrt{r_K^2-y^2-\left(r_Z^2-y^2-2x_{MK}\sqrt{r_Z^2-y^2}+x_{MK}^2\right)} \\ &\quad + (r_K^2-y^2)\arcsin\left(\frac{\sqrt{r_Z^2-y^2}-x_{MK}}{\sqrt{r_K^2-y^2}}\right) \\ &\quad + \sqrt{r_K^2-y^2}\sqrt{r_K^2-y^2-(r_K^2-y^2)} - (r_K^2-y^2)\arcsin\left(\frac{-\sqrt{r_K^2-y^2}}{\sqrt{r_K^2-y^2}}\right)\mathrm{d}y \end{aligned}$$

Hieraus folgt als Ergebnis das Integral:

$$\begin{aligned} V &= 2\int_0^a \left(\sqrt{r_Z^2-y^2} - x_{MK}\right)\sqrt{r_K^2-r_Z^2+2x_{MK}\sqrt{r_Z^2-y^2}-x_{MK}^2} \\ &\quad + (r_K^2-y^2)\left(\arcsin\left(\frac{\sqrt{r_Z^2-y^2}-x_{MK}}{\sqrt{r_K^2-y^2}}\right)+\frac{\pi}{2}\right)\mathrm{d}y \end{aligned} \quad (9)$$

Dieses Integral kann analytisch nicht gelöst werden. Eine weitere Lösung kann nur mit Hilfe eines numerischen Verfahrens erfolgen.

Überprüfung der analytischen Berechnung

Zur höheren Sicherheit in die Berechnung wird zunächst der folgende Integrand an zwei Stellen überprüft:

$$\begin{aligned} f(y) &:= \frac{1}{2}\left(\sqrt{r_Z^2-y^2} - x_{MK}\right)\sqrt{r_K^2-r_Z^2+2x_{MK}\sqrt{r_Z^2-y^2}-x_{MK}^2} \\ &\quad + \frac{1}{2}(r_K^2-y^2)\left(\arcsin\left(\frac{\sqrt{r_Z^2-y^2}-x_{MK}}{\sqrt{r_K^2-y^2}}\right)+\frac{\pi}{2}\right) \end{aligned} \quad (10)$$

Diese Funktion berechnet für jede Stelle $y \in [0, a]$ die Schnittfläche parallel zur xz-Ebene. Für y = a ist diese Fläche 0. Das Einsetzen in (10) liefert mit (5):

$$\begin{aligned} f(a) &:= \frac{1}{2}\left(\sqrt{r_Z^2-a^2} - \sqrt{r_Z^2-a^2} - \sqrt{r_K^2-a^2}\right)\sqrt{r_K^2-r_Z^2} \\ &\quad \overline{+2\left(\sqrt{r_Z^2-a^2}+\sqrt{r_K^2-a^2}\right)\sqrt{r_Z^2-a^2} - \left(\sqrt{r_Z^2-a^2}+\sqrt{r_K^2-a^2}\right)^2} \\ &\quad + \frac{1}{2}(r_K^2-a^2)\left(\arcsin\left(\frac{\sqrt{r_Z^2-a^2}-\sqrt{r_Z^2-a^2}-\sqrt{r_K^2-a^2}}{\sqrt{r_K^2-a^2}}\right)+\frac{\pi}{2}\right) \\ &= -\frac{1}{2}\sqrt{r_K^2-a^2}\sqrt{r_K^2-r_Z^2+2(r_Z^2-a^2)+2\sqrt{r_K^2-a^2}\sqrt{r_Z^2-a^2}} \\ &\quad \overline{-(r_Z^2-a^2)-2\sqrt{r_Z^2-a^2}\sqrt{r_K^2-a^2}-(r_K^2-a^2)} \\ &\quad + \frac{1}{2}(r_K^2-a^2)\left(\arcsin(-1)+\frac{\pi}{2}\right) \\ &= -\frac{1}{2}\sqrt{r_K^2-a^2}\sqrt{0} + \frac{1}{2}(r_K^2-a^2)\left(-\frac{\pi}{2}+\frac{\pi}{2}\right) = 0 \end{aligned}$$

An der Stelle y = 0 ermittelt der Integrand nach den Abbildungen (1) und (2) den Flächeninhalt eines halben Kreisabschnitts (nur y xxx 0) der Höhe *eK* + ***e z*** und des Radius *r K* , der nach [l], S. 145 berechnet wird:

$$A_{hA} = \frac{r_K^2}{4}(\alpha - \sin(\alpha)) \text{ mit } \alpha = 2\arcsin\left(\frac{2A}{2r_K}\right) = 2\arcsin\left(\frac{A}{r_K}\right)$$

oder

$$\begin{aligned} A_{hA} &= \frac{r_K^2}{4}(2\beta - \sin(2\beta)) \qquad \text{mit} \qquad \beta = \arcsin\left(\frac{A}{r_K}\right) \\ &= \frac{r_K^2}{2}(\beta - \sin(\beta)\cos(\beta)) \\ &= \frac{r_K^2}{2}\left(\arcsin\left(\frac{A}{r_K}\right) - \frac{A}{r_K}\sqrt{1-\left(\frac{A}{r_K}\right)^2}\right) \\ &= \frac{r_K^2}{2}\arcsin\left(\frac{A}{r_K}\right) - \frac{A}{2}\sqrt{r_K^2 - A^2} \end{aligned}$$

Diese Formel wird in der numerischen Berechnung mit ausgewertet (halber Kreisabschnitt AhA) und kann mit dem Wert des Integranden an der Stelle 0 verglichen werden.

Numerische Berechnung des Integrals

Zur numerischen Auswertung obiger Formel wähle ich das Rombergverfahren ([2], S. 108), das mit achtfacher Intervallhalbierung ausgeführt wird. Als Eingabe werden nicht die Radien sondern die anwendungsfreundlicheren Durchmesser benutzt.

Als Zahlenwerte für eine Berechnung werden die Angaben in Tabelle 2 benutzt:

Größe	Variable	Radius
Zylinder	*rz*	2.5 mm
Bohrloch	*a*	0.12 mm
Kugel	*TK*	15mm

Tabelle 2: Zahlenwerte

Ein Testprogramm, in das alle Größen in der Einheit Millimeter eingegeben werden, gibt zunächst interessante geometrische Größen aus.

KugZyl	30 5 .24
tangentialerHalbmesser:	0.12
tangentialer Durchmesser:	0.24
Radius Kugel:	1 5
Radius Zylinder:	2.5
Kugeldurchdringung:	0.002881661
Zylinderdurchdringung:	0.0004800077
Gesamtdurchdringung:	0.003361668474
axialer Halbmesser:	0.3175512
axialer Durchmesser:	0.6351024
2. Ber. axial.Halbmesser:	0.3175512
x-Koord. Kugelmittelpunkt:	17.49664
Integrand anderstelle 0:	0.0007116838
Integrand (tang. Halbm.):	0
halberKreisabschnitt AhA:	0.0007116838

Die zur Uberprüfung durchgeführte zweifache Berechnung des axialen Halbmessers und der Vergleich des Integranden and der Stelle 0 mit der halben Kreisabschnittsfläche zeigen jeweils die gewünschte Gleichheit.

Anschließend berechnet eine Romberg-Integration der Stufe 8 mit dem Integranden (10) ein Viertel des Volumens (siehe Tabelle 3). Wie aus der Berechnungstabelle hervorgeht, beträgt das Volumen näherungsweise 0.000201mm³.

.000042701	.000051221	.000050395	.000050325	.000050314	.000050313	.000050312	.000050312
.000049091	.000050447	.000050326	.000050315	.000050313	.000050312	.000050312	.000000000
.000050108	.000050334	.000050315	.000050313	.000050312	.000050312	.000000000	.000000000
.000050277	.000050316	.000050313	.000050312	.000050312	.000000000	.000000000	.000000000
.000050306	.000050313	.000050312	.000050312	.000000000	.000000000	.000000000	.000000000
.000050311	.000050312	.000050312	.000000000	.000000000	.000000000	.000000000	.000000000
.000050312	.000050312	.000000000	.000000000	.000000000	.000000000	.000000000	.000000000
.000050312	.000000000	.000000000	.000000000	.000000000	.000000000	.000000000	.000000000

Volumen: 0.0002012494748

Tabelle 3: Romberg Verfahren der Stufe 8 (Berechnungstabelle)

Grobe Überprüfung der numerischen Integration

Diese Berechnung wird größenordnungsmäßig überprüft. Das Volumen ist größer als ein Drittel des rechteckigen Quaders mit den Seitenlängen *2A: 2a* und der Gesamtdurchdringung $e_K + e_Z$, also größer als 4/3 · aA ($e_K + e_Z$). Dieser Ausdruck wird ebenfalls im Programm ausgewertet:

Vergleich 1/3 Quader: 0.0001708003

Auch diese Berechnung unterstützt das Ergebnis für das Volumen.

Das Programm Kugelzyl

Die Berechnung ist in einem Programm Kugelzyl zusammengefaßt, das als DOS-ausführbare Datei KugelzyLexe vorliegt. Es wird in der einfachsten Form unter Windows aus einer DOS-Box aufgerufen.

Der Name des Programms kann beliebig geändert, Laufzeitbedingungen und -wünsche können mit einer Windows-PIF-Datei angepasst, und der Aufruf auch durch ein Icon auf der Windows-Oberfläche vereinfacht werden.

Beim Programmablauf ist insbesondere auf die unterschiedlichen Einheiten der Eingabegrößen zu achten. Kugel- und Zylinderdurchmesser sind in Millimeter einzugeben: der kleine (tangentiale) Durchmesser des Bohrlochs dagegen in Mikrometer. Auch Dezimalzahlen mit Dezimalpunkt sind als Eingaben erlaubt.

```
        Berechnung des herausgeschliffenen Volumens                [CR]

Eingabe Kugeldurchmesser       [mm] > 30
Eingabe Zylinderdurchmesser    [mm] > 6
Eingabe kleiner Durchmesser 2a [µm] > 240

Ausgabe:
========
  Kalottentiefe (eZ + eK):     3.36 µm
  Volumen                  :  201249 µm^3
  großer Durchmesser 2A    :  636.10 µm

ENDE                                                              [CR]
```

Schlussbemerkung

Trotz aller Sorgfalt kann ich die Korrektheit der Ergebnisse nicht garantieren. Bei Abweichungen von zu erwartenden Ergebnissen müssen die Herleitung und der Programmcode überprüft werden.

Literatur

[1] J. Bronstein, K. Semendjajew: *Taschenbuch der Mathematik*, Teubner, Leipzig, 1969.

[2] J. Stoer: *Einführung in die numerische Mathematik I*, Springer, 1972.

Kontakt:

Dieter Kilsch,
Fachhochschule Bingen
kilsch@fh-bingen.de

Dr. Wolf Dehus

Opa und der neue Computer

(Fortsetzung aus APL-Journal 2/00)

Eigentlich war ich mit meinem alten Computer einigermaßen vertraut und daher auch zufrieden. Es fiel mir jedoch auf, daß in der letzten Zeit das Gesicht meines elfjährigen Enkels nach jedem Besuch, der gleichzeitig dem Computer galt, immer schiefer wurde. Ich versuchte, eine Erklärung zu finden, denn wer möchte schon einen Enkel mit einem schiefen Gesicht haben?

Also blieb mir nichts weiter übrig, als nach der Ursache des schiefen Gesichtes zu forschen. Das war nicht schwierig, denn bei einem Gespräch mit ihm, sozusagen unter Männern, erklärte er mir, daß es kaum noch Besitzer eines Computers gäbe, der ohne CD-Rom-Laufwerk ist. Nun erzählte er mir von dem Opa seines Freundes, der ein modernen Computer besaß, von den andern Großvätern, die überhaupt keinen hatten, sprach er natürlich nicht.

Mit schönen Worten, die meine Seele zunehmend berührten, wie: „Eigentlich bist du doch ein moderner Großvater“ und „Ich bin richtig stolz, wenn du bei meinen Schularbeiten mitreden kannst“ und „Meine Klassenkameraden finden dich prima,“ fühlte ich mich immer mehr wegen das alten Computers in die Enge getrieben.

Schon bald spürte das mein Enkel und nun versuchte er, das Eisen zu schmieden, solange es noch heiß ist, indem er direkt über die Vorteile und nahezu Wunder eines neuen Computers sprach. Als er meine schwindende Unentschlossenheit zum Erwerb eines neuen spürte, bot er mir an, mit seinem Taschengeld auszuhelfen, um sozusagen einen Trumpf auszuspielen.

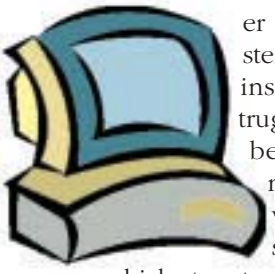
Zwar hatte er noch nicht meine Zusage, aber bereits einen Zettel in der Hand, auf dem er alles aufschrieb, was man beim Kauf eines neuen Computers beachten sollte. Nun teilte er mir vorsichtig mit, wo ein guter Laden ist; mit anderen Worten, es war ihm wieder mal gelungen, den Opa um den Finger zu wickeln. Also rief ich meinen alten Freund an, um jemanden beim Verhandeln dabei zu haben und verabredete mich mit ihm nach seiner Zusage zum Kauf des neuen Computers.

Als ich meinem Enkel am nächsten Tag telefonisch den Erwerb mitteilte, war er noch bevor ich den Hörer richtig aus der Hand legen konnte, bei uns. Als er sich jedoch die Hardware anschaute, wurde diesmal sein Gesicht nicht schiefer sondern immer länger. Worte wie: “Das ist doch keine Festplatte, höchstens ein Spielplatz für Flöhe“ oder “Die Frequenz kann man vergessen, die war schon zu Urgroßvaters Zeiten höher“, bzw. „Das ist doch eher eine Krankheit als ein Monitor, da war die Mattscheibe der Kamera obscura noch besser“ murmelte er so vor sich hin, daß ich sie gerade noch verstehen konnte.

Nun bezog er mit lieben Worten die noch liebere Oma ins Gespräch, die natürlich voll seine Ansicht vertrat, als wäre sie der Computerexperte, bis mir beide kategorisch rieten, das Gerät umzutauschen. Ganz nebenbei fragte mich mein Enkel aber wieder, bevor wir losfuhren, ob er noch etwas Taschengeld mitnehmen soll? Was das bedeutete, war nicht schwer zu erraten.

Als wir mit den Paketen in den Laden kamen, entstand für mich eine unangenehme Situation, die aber unser Sprößling mit fachmännischen Worten schnell überspielte und so den alten Kauf mit Hilfe eines neuen rückgängig machte, wobei die Hardware nicht schwerer, mein Portemonnaie dagegen erheblich leichter wurde.

Der Aufbau des Gerätes geschah dann unter seiner Obhut, ab und zu bat er mich, einen Beitrag zu leisten, der ihm aber stets „leicht“ korrekturbedürftig erschien. Ein Anruf seiner Eltern erinnerte ihn dann aber daran, daß es neben dem Computer auch Schularbeiten und andere Aufgaben gibt.



Am Wochenende fand er Zeit, die wichtigsten Programme zu installieren. Damit trug er gleich dazu bei, daß ich eine neue Vorstellung vom Begriff „Installation“ bekam und ich staunte, was alles in dem Stall des neuen Computers Platz hatte. Dabei sorgte er durch einige wohl ausgewählte Programme, daß auch ich nahezu computersüchtig wurde.

Einige Tage später beklagte sich mein Sohn bei uns darüber, daß unser braver Sprößling zu viel Zeit auf Kosten anderer Dinge dem Computer widme. Gleichzeitig schlug er mir vor, eine Kindersperre in den Computer einzuprogrammieren und beschrieb mir auch gleich die nötigen Schritte. Dann teilte er uns mit, daß unser Enkel wahrscheinlich auf dem Weg zu uns sei und ich schnell handeln sollte.

Das tat ich auch, allerdings erreichte unser Enkel unsere Wohnung eher und beobachtete mich lustig, wie ich versuchte, den Computer zu sperren; wobei es ihm nicht in den Sinn kam, daß diese Kindersperre ihm gelten sollte. Schließlich erklärte er mir, daß man das anders mache und schon war auf dem Monitor zu lesen, daß der Computer für Kinder außer Funktion sei. Jedoch ehe ich den Raum richtig verlassen konnte, war das Gerät wieder funktionstüchtig. Ich erklärte ihm schließlich, daß auch andere Dinge zu erledigen seien. Dem stimmte er dann brav zu und schon bald verabschiedete er sich von uns.

Da noch etwas Zeit bis zum Abendbrot war, wollte ich nun die vorhandene Pause für den Computer nutzen und schaltete das Gerät nochmals an.

Doch siehe da, nun strahlte mich der Monitor mit der Mitteilung an, daß das Gerät jetzt für Kinder gesperrt sei. Ich war empört, daß mein eigener Computer mich als Kind behandelte. Was sollte ich nun tun?

Zwar hätte ich bei meinem Sohn anrufen können, aber da meine Schwiegertochter mit Hochachtung im Bekanntenkreis erzählte, daß ihr Schwiegervater perfekt mit dem Computer umgehen könne, wollte ich diese, wenn auch irrige Vorstellung nicht zerstören. Es blieb mir also nichts weiter übrig, als auf den Computer zu verzichten und den nächsten Besuch meines Enkels abzuwarten.

In den kommenden Tagen staunte mein liebes Eheweib, daß ich mich schon eine Weile nicht mehr mit dem Computer beschäftigt hatte. Ich täuschte nun die Notwendigkeit anderer Beschäftigungen vor und wartete sehnlichst auf den Besuch meines Enkels.

Endlich klingelte es wieder einmal bei uns. Mit Freude im Gesicht stand unser Enkel an der Tür. Diesmal hatte er erst einmal Einiges der Omi zu erzählen, während ich sehnlichst darauf wartete, daß er zum Computer geht. Aber er hatte es zu meinem Leidwesen nicht so eilig. Schließlich riß doch mein Geduldsfaden, und ich fragte ihn, wie ich die „Kindersperre aufheben könnte.“ Darauf sagte er mir, daß ein Passwort eingegeben werden müsse, dieses stünde auf einem Zettel unter dem Mousepad. Als er, ohne den Computer zu beachten, nach Hause ging, fand ich kaum Zeit zur Verabschiedung und eilte ungeduldig zum Computer. Dort fand ich einen Zettel mit dem Wort „Opah“. Also „Opa“ falsch geschrieben! Natürlich gab ich nun die richtige Schreibweise von Opa in den Computer ein. Es passierte trotzdem nichts. Also unterlag ich bis zum nächsten Besuch wieder der Computerabstinenz. Das ging nun so weit, daß sich meine liebe Ehefrau Sorgen um meine Gesundheit machte.

Schließlich kam der Tag, an dem unser Enkel uns wieder mal besuchte. Als er die Sperre auf dem Monitor sah, erklärte er mir, daß doch eigentlich keine Kinder an den Computer kommen und diese Sperre damit unnötig sei, wobei er sich selbst sicher schon als Erwachsener sah. Er sprach mir aus dem Herzen (allerdings aus anderen Gründen) und löschte „mit meiner stillen freudigen Zustimmung“ die Kindersperre.

Auf meinen Hinweis, daß man Opa ohne „H“ schreibt, erklärte er mir, daß das Paßwort stets 4 Buchstaben haben müsse und er deshalb ein „H“, sozusagen von „Honor / Ehre“ abgeleitet, zusätzlich zu Opa eingesetzt habe (Dabei verstand er, gleichzeitig auf seine ersten Lateinstunden hinzuweisen). Etwas beleidigt fragte er, ob ich ihn für so dumm halte?!

Er beschäftigte sich noch eine Weile mit dem Computer, aß mit uns Abendbrot und ging dann, nachdem er sich von uns herzlich verabschiedet hatte, nach Hause. Mir sagte er noch beiläufig, daß er den Computer diebstahlsicher gemacht hätte. Darüber konnte ich nur lächeln, ohne zu ahnen, was er damit meinte.

Ungeduldig, aber glücklich über diesen Tag, an dem ich sehnlichst hoffte, endlich wieder den Computer nutzen zu können, verabschiedete ich mich von ihm und setzte mich endlich wieder an den Computer. Doch nichts lief, Mein lieber Enkel hatte nun, um ihn diebstahlsicherer zu machen, statt der Kindersperre die Benutzersperre einprogrammiert.

Ist es nun verständlich, daß ich zum Verächter der Computer geworden bin?

Nina Kroll

Frauen und Computer

Donnerstagabend, 20.45 Uhr. Eigentlich wollte ich zum Tennis, mich ein wenig von dem Streß in der Uni ablenken, doch es regnete mal wieder in Strömen, so daß mir nichts Besseres einfiel, als den Computer einzuschalten.

Ich bin Studentin an der TU Berlin und studiere Informatik im 4. Semester. Im Volksmund werden wir auch Computerfreaks genannt, wen wundert es, daß ich, kaum den Computer in der Uni abgeschaltet, ihn zu Hause wieder anschalte?

Ich hatte natürlich noch einige Dinge für die Uni zu erledigen, eine Ausarbeitung zu Tabellenverarbeitung in Access und Hausaufgaben in „struktureller und diskreter Mathematik für Informatiker". Aber so richtige Lust verspürte ich keine, also beschloß ich, meine Festplatte nach irgendwelchen Dateien zu durchsuchen, die ich nicht mehr brauchte. Ich fing munter an, Dateien und Ordner zu löschen, hier und da war auch ein veraltetes Programm an der Reihe....

Nach circa einer dreiviertel Stunde, war ich der Meinung, ich könnte nun die neuesten Versionen der veralteten Programme installieren. Leider hatte mein Computer eine andere Auffassung und mein CD-ROM-Laufwerk kapitulierte. Gegen 0.30 Uhr gab auch ich mich nach zahlreichen vergeblichen Versuchen geschlagen und brach über meiner Tastatur auf dem Schreibtisch zusammen. Meine letzten Gedanken, die ich vor mich hin murmelte, bevor ich die Kiste ausschaltete, waren: „Frauen und Computer!

Vor einigen Wochen bat mich Martin Barghoorn, Redakteur dieser Zeitschrift, einen Bericht aus „unserer" Sicht über *Frauen und Computer* zu schreiben. Dieser Donnerstagabend gab mir einen guten Anlaß, den Artikel zu verfassen.

Kann man meinen Vorfall verallgemeinern? Kann man ihn auf unsere ganze Gattung beziehen? Sind Aussprüche berechtigt wie: „J*a, ja, Frauen und Technik!"* oder „*Um Gotteswillen, Frauen und Computer!"?* - Natürlich nicht.

Dennoch ist der Frauenanteil in naturwissenschaftlichen und technischen Fächern sehr gering. Schon in der Schule sind es die Jungs, die in Mathe, Chemie und Physik glänzen und meist zu Hause an ihren Computern herumbasteln.

Doch besteht nicht die Möglichkeit, auch von den Frauen etwas zu lernen? Könnte es nicht sein, das Frauen ganz andere Gedankengänge aufweisen und somit eventuell neue Ideen entwickeln? Um das auszuprobieren, müßte man eine ganze Reihe von Frauen zur Verfügung haben. Doch wie könnte man den Frauen ein naturwissenschaftliches oder technisches Fach wie Informatik oder Elektrotechnik schmackhaft machen?

Wenn es die Angst, das Studium nicht zu schaffen, sein sollte, die Frauen von diesen Fächern fernhält, dann kann ich nur berichten, daß auch viele männliche Informatikstudenten ihr Studium abrechen. In meinem Jahrgang gibt es viele Kommilitonen, für die es gilt, sich durchzubeißen – gerade im Grundstudium. Die Durchfallquoten in den Klausuren sind sehr hoch. Wir Frauen würden deshalb nicht auffallen, wenn wir dabei wären. Man muß die Zähne zusammenbeißen und durchhalten.

Zugegeben: Bei einigen Frauen existiert ein Desinteresse gegenüber technischen Sachen. Dieses Desinteresse darf man jedoch nicht auf die Gesamtheit beziehen.

Die IT-Branche wird nach wie vor von Männern bestimmt – und daran wird sich in der nächsten Zeit nichts ändern. „Frauen und Computer", das ist immer noch eine Rarität. Ein Grund für Unternehmen, Frauen einzustellen, dann haben sie etwas Besonderes, etwas Eigenes (wie Loriot es ausdrücken würde).

Kontakt:

Nina Kroll, nkroll@cs.tu-berlin.de

Allgemeine Informationen

(Stand 1. Dezember 1999)

Vorstand

Vorsitzender:

Dieter Lattermann
Rheinstraße 23
69190 Walldorf
Tel. (06227) 63469
E-Mail: dieter_lattermann@compuserve com

2. Vorsitzender:

Martin Barghoorn
Technische Universität Berlin
Sekr. FR 6-9
Franklinstr. 28
10587 Berlin
Tel. (030) 314 24392
Fax: (030) 314 25901
E-Mail: barg@cs.tu-berlin.de
Web: http://stat.cs.tu-berlin.de/~barg/

Schriftführer

Wolfgang Dreßen
McKinsey & Co.
Königsallee 60c
40027 Düsseldorf
Tel. (0211) 136-443
FAX: 0211/136-4997-4437
E-MAIL: wolfgang_dressen@McKinsey.com

Schatzmeister

Gert Osterburg
Am Hochholz 7
61184 Karben
Tel. (06034) 2995
E-Mail: Gert.Osterburg@T-Online.de

Beitragssätze

Ordentliche Mitglieder:

Natürliche Personen	60,00	DM	p.a.
Studenten / Schüler	20,00	DM	p.a.

Außerordentliche Mitglieder:

Juristische / natürliche Personen ≥ 1.000,00 DM p.a.

Bankverbindung: BVB Volksbank eG Bad Vilbel
BLZ 518 613 25, Konto-Nr. 523 2694

Hinweis:

Wir bitten alle Mitglieder, uns immer gleich *Adressenänderungen* und *neue Bankverbindungen* mitzuteilen.

Geben Sie bei *Überweisungen* den *Namen* und/oder die *Mitgliedsnummer* an.

Einzugsermächtigung

Ich erkläre mich hiermit widerruflich damit einverstanden, daß APL Germany e.V.

den jeweils gültigen Jahres-Mitgliedsbeitrag von meinem unten angegebenen Konto abbucht.

Einen eventuell bestehenden Dauerauftrag habe ich bei meiner Bank gelöscht.

Bankbezeichnung: _____

BLZ: _____ Konto-Nr.: _____

Datum: _____ Unterschrift: _____

APL Germany e.V. Aufnahmeantrag/Änderungsanzeige

APL Germany e.V.
Herrn Gert Osterburg
Am Hochholz 7

D-61184 Karben

❏ Ich ersuche / wir ersuchen um Aufnahme in den *APL Germany e.V.*

als außerordentliches Mitglied (❏ jur. Person ❏ natürl. Person)

Die Vereinssatzung ist mir / uns bekannt: Ja ❏ Nein ❏

❏ Änderungsanzeige

Der Verein darf Aufkleber mit meiner / unserer Postanschrift weitergeben:

Ja ❏ Nein ❏

	Dienstanschrift/ Hauptansprechpartner	Gegebenenfalls weitere Mitglieder der Organisation
Firma		
Abteilung o.ä.		
zu Hd. von		
Straße, Hausnummer		
Postfach		
PLZ, Ort		
Bei Ausland: Staat		
Telefon (Vorwahl / Ruf)		
FAX (Vorwahl / Ruf)		
E-Mail-Adresse		

Die Angaben dieses Formulars werden für vereinsinterne Zwecke auf Datenträger gespeichert.

Ort, Datum: _____ Unterschrift: _____

Lesen Sie unsere Anleitung im Internet: www.rhombos.de

Book on Demand

BOOK ON DEMAND

Ideal für die kostengünstige Publikation von wissenschaftlichen Berichten, Tagungsbänden, Gutachten, Monographien, Buchreihen, Diplom- und Doktorarbeiten ... inklusive Verlagsbetreuung!

Planen Sie, ein Buch zu veröffentlichen? Haben Sie vielleicht schon ein fertiges Manuskript in der Schublade, aber bisher vergeblich versucht, einen Verlag zu finden?

Der Weg, den ein Manuskript gehen muß, bis es in der Bücherauslage einer Buchhandlung landet, ist beschwerlich. Es sind nicht immer inhaltliche Gründe, die dazu führen, daß Verlage Manuskripte ablehnen. In erster Linie spielen hier finanzielle Erwägungen eine Rolle, denn der Aufwand, der für eine Buchproduktion betrieben werden muß, ist enorm. Entsprechend hoch ist das finanzielle Risiko für den Verlag, das mit einer Buchveröffentlichung einhergeht.

Vielfach bleiben auf diese Weise exzellente Bücher auf der Strecke, für die nur Kleinauflagen in Frage kommen, weil sie nur einen begrenzten Kreis von Abnehmern erwarten lassen. Besonders hart betroffen davon sind beispielsweise Fachbücher. Vielfach leisten Autoren deshalb erhebliche Druckkostenzuschüsse, um dennoch veröffentlicht zu werden.

Mit dem Book-on-Demand-Verfahren bieten wir hier eine Alternative. Wir können auf diese Weise den Druck der Bücher direkt an die Nachfrage koppeln und können Ihr vollwertiges Buch selbst in Kleinstauflagen schnell und preiswert publizieren. Gedruckt wird nur noch, was bestellt wird. Damit reduzieren wir die Druck- und Lagerkosten auf ein Minimum, entsprechend sinkt das Investitionsrisiko.

Bei dieser Herangehensweise setzen wir auf eine enge Kooperation mit unseren Autoren. Sie liefern uns Ihre Arbeit als Datei, am besten im Postscript-Format. Wie das geht, können Sie unserer Anleitung im Internet entnehmen. Anschließend geht Ihr Werk in den Digitaldruck und schon nach wenigen Tagen können Sie Ihr Buch in den Händen halten. Zu einem sagenhaft günstigen Preis!

Und selbstverständlich sind Sie an den Einnahmen beteiligt. Überzeugen Sie sich selbst und informieren Sie sich auf unserer Internetseite!

Wir führen die gesamte verlagstechnische Betreuung Ihres Werkes durch, hierzu zählen beispielsweise Vermarktung, Auslieferung und Abrechnung. Die gedruckten Exemplare erhalten eine ISBN-Nummer und sind damit weltweit über das Verlagsverzeichnis zu identifizieren. Zusätzlich wird Ihre Publikation mit Inhaltsverzeichnis und Kurzbeschreibung/Vorwort in den RHOMBOS-Online-Shop aufgenommen, über den sie bequem bestellt werden kann, auf Wunsch auch in elektronischer Form (PDF), beispielsweise aus dem Ausland, wenn der Versandweg zu lange dauern würde.

Interessiert? Wir beraten Sie gerne!

RHOMBOS-VERLAG, Kurfürstenstr. 17, D-10785 Berlin
Tel. 0930-261 68 54 eMail: verlag@rhombos.de