Stellingen

behorende bij het proefschrift

Adaptive
Computer-based Training in Engineering Education

Rik Essenius

Delft, 9 mei 1995
1 De tijdwinst die gebruik van de in het proefschrift beschreven oefenprogrammatuur oplevert aan de docent, heeft een niet-minimum-fase karakteristiek.

2 Voor oefenprogrammatuur bij technisch-wetenschappelijke vakken is een éédimensionaal adaptatiemechanisme, gebaseerd op de moeilijkheidsgraad van vraagstukken, vaak onvoldoende. Een tweede dimensie, gebaseerd op de taxonomie van het vak, biedt een goede aanvulling.

3 Het World Wide Web biedt uitstekende mogelijkheden om interactieve, platform-onafhankelijke instructieprogrammatuur aan te bieden.

4 Om het leereffect bij studenten aan technische universiteiten te bevorderen moet er meer gebruik gemaakt worden van onderwijsvormen die de student activeren, waar-onder computer-ondersteund onderwijs.

5 Gebruik maken van vraagstukken die herkenbare problemen beschrijven en van vraagstukken met een actieve inbreng van simulaties verhoogt het leereffect en verleendigt een oefensessie.

6 De term “positieve discriminatie” is, gezien de huidige betekenis van het woord discriminatie, een interne tegenstrijdigheid.

7 Persoonlijke vrijheid houdt op waar die van een ander in het gedrang komt.

8 Het feit dat vanuit het bedrijfsleven voor informatica-vacatures wordt geworven bij de Faculteit der Elektrotechniek, zou zowel de Faculteit der Technische Wiskunde en Informatica als de Faculteit der Elektrotechniek reden tot nadenken moeten geven.

9 De universiteiten houden onvoldoende rekening met het feit dat de meeste promovendi slechts in tijdelijke dienst zijn, waardoor veel expertise verdwijnt met de promovendus.

10 Het gebruik van de in het Engels synonieme woorden “expanded” en “extended” ter aanduiding van verschillende methodes voor geheugengebruik karakteriseert de ondoorzichtigheid van de architectuur van MS-DOS gebaseerde PC’s.

11 Omdat bij alle diabetespatiënten zelfcontrole het optreden van complicaties kan helpen voorkomen, zal het een positief effect hebben op zowel de kwaliteit van het leven als de totaal te vergoeden kosten om ook aan niet-insuline-afhankelijke diabetici middelen voor zelfcontrole te vergoeden.

12 De stelling “een informaticus is een veredelde programmeur” is vergelijkbaar met de stelling “een werktuigbouwkundige is een veredelde monteur”.

13 De verkeersveiligheid op verbindingswegen zou sterk worden bevorderd door het asfalteren van ernaast gelegen fietspaden.

14 Het duidelijkste gevolg van het toegenomen aantal zenders op het televisienet is dat het nu langer duurt voordat je zeker weet dat er niets op TV is.
1. For the teacher, the gain in time produced by using the training software as described in the thesis has a non-minimum-phase characteristic.

2. For training software used in engineering courses, a unidimensional adaptation mechanism, based on the difficulty level of the questions, is often not sufficient. A second dimension, based on the course taxonomy, offers a good addition.

3. The World Wide Web has excellent possibilities to offer interactive, platform-independent training software.

4. To increase the learning effect with students at technological universities, types of education that activate the student, among which computer-assisted instruction, should be used more often.

5. Making use of questions describing recognisable problems and of questions with an active contribution of simulations increases the learning effect and makes a training session more lively.

6. Considering the current connotation of the word discrimination, the term “positive discrimination” is a contradiction in terms.

7. Personal freedom ends where that of another may well suffer.

8. The fact that several companies recruit for their computer science vacancies at the Department of Electrical Engineering, should give cause for thought to both the Department of Technical Mathematics and Informatics and the Department of Electrical Engineering.

9. The universities do not take sufficiently into account that most Ph.D. students are only temporarily employed, through which much expertise disappears with the Ph.D. student.

10. The use of the synonymous words “expanded” and “extended” to indicate different methods of memory use characterises the uncleanness of the architecture of MS-DOS based PCs.

11. Because for all diabetes patients self-checks can help to prevent the occurrence of complications, there would be a positive effect both on the quality of life, and on the total cost to be defrayed, if the means for self-checks were financed for non-insuline-dependent diabetics as well.

12. The proposition “a computer scientist is an ennobled programmer” is comparable to the proposition “a mechanical engineer is an ennobled mechanic”.

13. Traffic safety on connecting roads would be vastly improved by asphalting the bike-ways alongside.

14. The clearest consequence of the increased number of channels on the television network is that now it takes a longer time before you are certain there is nothing on TV.
Adaptive Computer-based Training in Engineering Education

Rik Essenius
Adaptive Computer-based Training in Engineering Education

PROEFSCHRIJT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op dinsdag 9 mei 1995 te 10:30 uur

door

Richard Peter ESSENIUS

ingenieur informatica

geboren te Rotterdam
Dit proefschrift is goedgekeurd door de promotoren:

Prof.ir. G. Honderd
Prof.dr. W.M.G. Jochems

Samenstelling promotiecommissie:

Rector Magnificus
Prof.dr.ir. K.L. Boon
Prof. F. de Coulon
Prof.ir. G. Honderd
Prof.dr. W.M.G. Jochems
Prof.dr.ir. J. van Katwijk
Prof.drs. C.F. van der Klauw
ir. C. Wissenburgh

Technische Universiteit Delft
Open Universiteit Heerlen
Ecole Polytechnique Fédérale de Lausanne
Technische Universiteit Delft
Technische Universiteit Delft
Technische Universiteit Delft
Erasmus Universiteit Rotterdam
Technische Universiteit Delft

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Essenius, Richard Peter

Adaptive computer-based training in engineering education /
Richard Peter Essenius. - [S.l. : s.n.]. - III.
Proefschrift Technische Universiteit Delft. - Met index,
lit. opg. - Met samenvatting in het Nederlands.
ISBN 90-9008191-7
Trefw.: computergebruik in het hoger technisch onderwijs.
In memory of my uncle Cornelis Roodenburg

*When the sun stops shining, you start to see more and more stars*
This document was prepared with FrameMaker 4®

Cover design by Rik Essenius and André van der Ham

Printed by the Universiteitsdrukkerij Delft

© 1995 by Rik Essenius

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior consent of the author.
Acknowledgements

The research described in this thesis was performed at the Control Laboratory of the Department of Electrical Engineering, Delft University of Technology. There was close cooperation with the Educational Development Unit of the Department of Philosophy and Humanities, and the Electronics Laboratory of the Department of Electrical Engineering, at the same university. The project was funded by the Onderwijs Stimuleringsfonds (Education Stimulation Fund).

The work in this thesis could not have been carried out without the help of many people. First of all I express my gratitude to Ger Honderd and Wim Jochems for their support and encouragement during the research, while leaving me very much room for initiative. Wim Jongkind is gratefully acknowledged for his guidance as project manager during the first two years. Maarten van de Ven at the Educational Development Unit supported me greatly with the educational component of the research, the test setups and evaluation. Further, I want to thank Kees Wissenburgh, the project coordinator for Electronics, with whom I worked very closely in realising the test sessions. A special word of appreciation goes to Mrs. J.B. Zaat-Jones for correcting the English text.

My colleagues and ex-colleagues at the lab created a great working atmosphere. I want to thank especially Cornelis Klomp, André van der Ham, Edward Holweg, Marcel Brunt and Richard Koch for their friendship and support. Further I want to mention my international friends, in particular Irena Ivanova and Anne Manse. The physical distance did not stop our pleasant discussions.

Last but definitely not least I thank my parents and my brothers Edwin and Robin. I am obliged for their continuous support throughout the years.
Table of contents

Acknowledgements

Summary

1 Introduction

1.1 What makes instruction productive? ........................................... 1
1.2 The terms in the title .......................................................... 3
  1.2.1 Training ................................................................. 3
  1.2.2 Computer-based training ............................................... 4
  1.2.3 Adaptive computer-based training .................................. 5
  1.2.4 Training in engineering education .................................. 6
1.3 The project and its participants ........................................... 7
1.4 Synopsis of the thesis ........................................................ 8

2 Computer support in training and testing .................................. 9

2.1 Computer-assisted instruction .............................................. 9
  2.1.1 History ................................................................. 9
  2.1.2 Structures ........................................................... 10
  2.1.3 Control strategies .................................................... 11
2.2 Item banking ....................................................................... 12
2.3 Adaptive strategies in testing ............................................. 14
  2.3.1 The link between training and testing ............................. 14
  2.3.2 Test length adaptation ............................................... 15
  2.3.3 Test length and level adaptation: Item Response Theory .... 17
Table of contents

3 Artificial intelligence and computer-based training 31

3.1 Introduction .................................................. 31
3.2 Expert systems ................................................. 31
  3.2.1 First generation expert systems .......................... 32
  3.2.2 Second generation expert systems ....................... 34
3.3 Machine learning techniques ............................. 34
  3.3.1 What is learning? ........................................ 35
  3.3.2 Learning from examples ................................. 37
  3.3.3 Learning by observation ................................. 38
  3.3.4 Case-based reasoning ................................... 40
  3.3.5 Reinforcement learning ................................. 41
  3.3.6 Biologically inspired methods ......................... 43
  3.3.7 Discussion ................................................ 46
3.4 Blackboard architectures ................................. 46
3.5 Conclusion .................................................... 48

4 A general computer-based training shell 49

4.1 Purpose .......................................................... 49
4.2 Basic assumptions ............................................. 49
4.3 Design approach ............................................... 51
4.4 The analysis phase ............................................ 51
  4.4.1 The models ............................................... 52
  4.4.2 Control ................................................... 53
  4.4.3 The item interface protocol ............................. 54
  4.4.4 Platform-dependent objects ............................. 56
  4.4.5 The analysis model ...................................... 57
4.5 The design phase .............................................. 59
  4.5.1 The course model ....................................... 59
  4.5.2 The item manager ...................................... 62
6.4 Taiga extensions: software requirements specification .......... 96
  6.4.1 Introduction ........................................ 96
  6.4.2 General description ................................ 99
  6.4.3 Functional requirements of I^2T ....................... 101
  6.4.4 Functional requirements of GT^2 ...................... 103
  6.4.5 Functional requirements of CT^2 ...................... 104
  6.4.6 External interface requirements ....................... 108
  6.4.7 Performance requirements ............................ 109
  6.4.8 Attributes ........................................... 109

6.5 Design of the Items-D type Interface for Taiga ................. 109
  6.5.1 High-level structure ................................ 109
  6.5.2 Function refinements ................................ 111

6.6 Design of the Control Toolbox for Taiga ....................... 117
  6.6.1 Interfacing with the environment ...................... 117
  6.6.2 Display of time responses ............................ 118
  6.6.3 Display of pole-zero configurations ................... 121
  6.6.4 Display of frequency responses ....................... 123
  6.6.5 Display of Mason-type flow diagrams .................. 127

6.7 Design of the General Toolbox for Taiga ..................... 130
  6.7.1 Display of external raster images ..................... 130
  6.7.2 Display of external texts ............................ 133

6.8 Example items in Taiga ................................... 134

6.9 Porting the toolboxes to other environments .................. 136

6.10 On-line application of simulation software .................. 136

6.11 Conclusion ............................................. 139

7 Experiments and results .................................................. 141

7.1 Test set-ups ................................................. 141
  7.1.1 Item characteristics ................................ 141
  7.1.2 Test situation ....................................... 143
  7.1.3 Functionality test of the implementation environment ... 144
  7.1.4 Functionality test of the adaptive system ............. 145
  7.1.5 The questionnaire with the second test ................ 145

7.2 Statistical analysis of adaptive system test ................ 147
  7.2.1 Introduction ....................................... 147
  7.2.2 Global indication of questionnaire result ............. 147
7.2.3 Questionnaire analysis .............................................. 149
7.2.4 Analysis of session results ......................................... 152
7.2.5 Analysis of the items asked ...................................... 155
7.3 Qualitative analysis of the test results ......................... 157
  7.3.1 Technical performance ........................................ 157
  7.3.2 The evaluation discussions: student judgements ........ 158
  7.3.3 Lecturer and observer experiences ......................... 160
  7.3.4 Examples of adaptation paths ............................. 162
7.4 Conclusions .................................................................. 163

8 Conclusions and recommendations ................................. 165
  8.1 Research outcomes ................................................ 165
  8.2 Project reflection .................................................. 167
  8.3 Recommendations ................................................ 169
    8.3.1 Technical recommendations ............................ 169
    8.3.2 Organisational recommendations .................... 169
    8.3.3 Educational recommendations ....................... 171

References ...................................................................... 173

Abbreviations .................................................................... 183

A Format specifications .................................................. 185
  A.1 The item information file DATA.TDL .................... 186
    A.1.1 Syntax ...................................................... 186
    A.1.2 Semantics ................................................ 187
    A.1.3 Example .................................................. 188
  A.2 The item result file RESULT.TDL ....................... 189
    A.2.1 Syntax ...................................................... 189
    A.2.2 Semantics ................................................ 190
    A.2.3 Example .................................................. 191
  A.3 External text file format of the General Toolbox ......... 191
    A.3.1 Description ................................................ 191
    A.3.2 Discussion ................................................ 192
<table>
<thead>
<tr>
<th></th>
<th>The questionnaire</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>The original Dutch version</td>
<td>194</td>
</tr>
<tr>
<td></td>
<td>The English translation</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>Questionnaire result data</td>
<td>198</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Using the World Wide Web for training applications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Introduction</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>The principle of WWW</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>User input with WWW</td>
<td>202</td>
</tr>
<tr>
<td></td>
<td>Discussion of usability</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td>Needed system modifications for WWW usage</td>
<td>203</td>
</tr>
</tbody>
</table>

Adaptieve instructie per computer in technisch-wetenschappelijk onderwijs 205

Curriculum Vitae 209

Index 211
Summary

Training is considered an important element in learning engineering courses. For the training to be effective, there must be active participation by the students. To achieve this, the group size should be small in order to be able to give adequate feedback to the student and to provide the student with exercises matching his/her level of mastery. However, the usually limited human resources prohibit giving training to small groups. A computer-based training system can provide a solution for this dilemma. For maximum productivity, the system must adapt training to the student’s level. The aim of the research is to realise an adaptive training system for engineering education.

Computer-assisted instruction programs have been around since the early sixties. Although often quite sophisticated, the majority of the early systems were rigid and they depended heavily on what the designer had anticipated. From the late seventies, other possibilities were explored and there was research into intelligent tutoring systems and item banking, to name only a few areas. Item banking is usually used for testing purposes. The teacher describes a subset of a data base of questions (called items), and lets the system create several sessions from the subset. The student is administered one of these sessions. Although this scheme can also be used for training, an adaptive method is preferred. A method is called adaptive if it is able to select an item on the basis of the student’s performance on earlier items. Because courses in engineering domains are usually multidimensional, i.e., they do not depend on a single underlying trait, a two-level adaptation is here proposed: a course level adaptation to select a subject, and a subject level adaptation to select an item of the chosen subject. In intelligent tutoring systems research, student models and curricular models are used to adapt to the students. These models are applicable in an adaptive training system as well. The suggestion is to use these models, but to leave the domain knowledge inside the items, providing a more general level of usability.
Summary

With a view to possible application in the software to be developed, the usability of several artificial intelligence techniques has been investigated. Machine learning techniques do not seem suitable for a training system as intended in this project. For the system architecture, a blackboard system (which consists of independent agents using a shared database) can provide a flexible basis.

The implemented software for a general adaptive session manager consists of a shell that allows the use of different adaptation methods. It is based on the two-level adaptation suggested earlier. The actual implementation of the adaptation methods is unspecified in the shell. A requirement of the course is that it can be represented as a taxonomy of subjects or concepts which form knowledge units that can be trained or tested separately. An object-oriented method is used, specifically to support encapsulation. The system was implemented in C++ to enable portability.

For the pilot courses, the course level model consists of an and-graph with prerequisite relations; the subject level model is based on item response theory (IRT). The IRT method of selecting an item that gives the most information assumes that the student's mastery level is constant. Further, it can lead to undesirably large jumps in the level of difficulty. Therefore a heuristic level update process is applied, that gives a more gradual change and allows variations in the student's mastery level.

The project's two pilot courses are introductory courses on Control Theory and on Electronics. For item implementation, the authoring system Taiga was used most frequently. To make Taiga suitable, an interface with the management system was built, as well as a toolbox to provide control theory related graphs, and a toolbox to provide the display of external images and texts.

The system has been tested in the Electronics course with a group of 29 students. The lecturer implemented the items himself in a reasonably short period, despite some drawbacks of using Taiga for such an application. The sample group was not large enough to make definitive statements, but the course lecturer judged that the system performed quite well. Further, the system was appreciated by most students. Naturally, there was criticism as well, mostly on the fact that the students did not have any influence on the decision process, and that they lacked a study advice.

In conclusion, the main aim of the research has been achieved. A generally applicable training system is now available. However, more tests are recommended before using the system in practice. In addition, it is advised to implement a more advisory adaptation method in order to enable the students to have more influence. Further, a method to refine the item parameters, based on session results, and a study advice generator would be welcome extensions. A final recommendation is to make the system available via the World Wide Web for increased availability concerning time, place and platform.
Introduction

Everybody knows the stereotypic image of the professor giving lectures in very large lecture theatres. Although it is a common way of transferring knowledge at a Dutch university, especially in the first two years, it is very unilateral. Communication from the students to the lecturer is virtually impossible. Students in the propaedeutic year comprise a very large and heterogeneous group, and such a group is difficult to teach [Guskey, 1988]. The dilemma the teachers have to deal with is that of how to address a large number of students while using limited human resources, and also guaranteeing the quality of the education. In other words, the instruction has to be productive. The first section of this chapter summarises factors that make instruction productive.

In engineering education, training has been found to be very important. The title of this thesis consists of the four terms training, computer-based, adaptive and engineering education. They are explained in section 1.2. The research must be seen as a design project aimed at a usable product, namely an adaptive computer-based training program for courses in electrical engineering. The project is described in section 1.3. An overview of the contents of the thesis is given in section 1.4.

1.1 What makes instruction productive?

There has been much research on educational productivity [Fraser et al., 1987]. A well-known model of student learning is defined by Glaser [Glaser, 1977; 1982]. This model is primarily related to learning processes, and concentrates on procedures for effective learning. Glaser identified four components considered essential for student learning:

1. Analysis of competent performance (anticipated outcomes).
2. Description of the learner's initial state.
3. Transformation process between the initial state and a state of competence.

4. Assessment of the effects of instructional implementation.

The third component is the unique contribution of the Glaser-type models. This component requires study activity from the student: studying for an exam, following lectures, doing practicals, doing exercises, making reports and study planning. There is no direct relation between instruction and learning results. Using the offered instruction, the student develops study activities which may lead to certain results. This is shown schematically in figure 1.1 [Jochems, 1990], in which the arrows give the main direction of influence. For the students, two factors appear to

![Diagram](image)

*Figure 1.1. The relation between learning results and instruction*

be of major importance for the achievement of learning results: ability and effort. In engineering education, the important abilities concern mathematics and physics. It is a characteristic of engineering education that effort is especially the amount of time spent on training.

To reach a certain result, both effort and ability are needed, but a less talented student can compensate for such a lack by putting in more effort, and a talented student will, correspondingly, put in less effort for a comparable learning effect. It is plausible to argue that increasing the students' efforts, making them study more, will increase the chance of success, other things being equal.

Research has shown that the amount of instruction influences the learning result considerably, and this factor can be influenced by the institute. However, it is likely that the time needed to follow more instruction will be withdrawn from home study. Further, increasing the amount of instruction is only useful if the quality of instruction is sufficient. In a quantitative research study it was concluded that the improvement of learning results is realised by giving study advice to the students, and encouraging active student participation by means of assignments and training, in combination with adjustments and progress supervision [Lysakowski & Walberg, 1982]. Fraser concluded that in the learning process, the most important factors are considered reinforcement, remediation and feedback, mastery learning, tutoring and
1.2 The terms in the title

A summary of the basic problem to be dealt with is: given the unilaterality of the most common teaching form in higher education, there is only a restricted type of learning activity, which causes other learning activities to be underrepresented. The most important problem is that the students are usually too passive during instruction. In the Netherlands, students have much freedom in choosing whether or not to prepare and to follow the course lectures, so much initiative is left to themselves. From their study behaviour, it appears that students are not always able to cope with this freedom. Since we are not able to provide the students with the most productive learning environment, we will have to stimulate active participation.

1.2.1 Training

In section 1.1 it was shown that in order to achieve productive instruction, active student participation is needed. Students should try to solve problems by themselves. However, it is not likely that the current system of large-scale lectures will change in the near future. At the moment, the most common way of teaching courses is by giving lectures to the entire group, and by doing separate training sessions with smaller groups of about 40 students. Sometimes also practicals are included but, because of the large groups, these are often divided throughout the year, and therefore many practicals not to coincide with the instruction period.

Despite the measures taken, students often postpone studying until just before the exam, and they are not very active even during the training sessions. It can be noticed very often that students just write down the solution the instructor writes on the blackboard, to study it later. But even being able to follow what the teacher has done is completely different from finding the solution and the required steps yourself. Some teachers try to activate the students by handing out assignments after
each lecture, which have to be handed in at the next lecture, but this takes a lot of time and still involves the risk that students copy the answers from each other.

Thus a 'personal trainer', who encourages the student to solve the problems and offers help only when necessary, would be ideal. However, the large number of students restricts the possibility of personal instruction, as there are only a few instructors available.

1.2.2 Computer-based training

A possible solution to the dilemma described in the previous section is provided by application of computer-based training (CBT). In this thesis, the term CBT is used for activities in which the computer presents a problem, registers answers, gives feedback and cues when necessary, and presents a next problem (either by choosing the problem itself or by letting the student choose).

Seen from both the teacher's and the student's point of view, computer-based training can have many benefits. However, we should be careful not to be too optimistic. History has shown that very often projects have failed for various reasons. Possible advantages and drawbacks of CBT are:

individual orientation. Students can work at their own pace. Slower students are not rushed to keep up with the group, and fast students can go as fast as they want. Computer materials are inherently self-paced.

availability. A computer can be used whenever wanted. However, that it can be used is not to say that it will be used. If left completely voluntary, the reward for using the computer must be clear, otherwise it will probably not be used at all.

patience. A computer does not get bored or upset after repeating the same thing over and over again. Care should be taken, however, that the students do not get bored.

non-humanness. Students seem to be less reluctant in approaching a computer than in approaching the teacher [Pullen & Mercer, 1988]. This non-humanness is also a drawback. Humans are capable of seeing the state of the students and making all kinds of connections a computer would never be able to make. For example, most teachers can estimate a student's level within a few minutes. Humans can adapt to unknown situations, and see problems outside a given context. Computers only perceive a limited image of the student. This is known as the bandwidth problem [Goldstein, 1982; cited by Bieman, 1991].

dynamic representations. The strength of a computer is its computational ability. A computer can execute and show interactive simulations to enhance the compre-
hension of difficult course matter. For example, given a model of an epidemic, it could be possible to allow the students to change model parameters such as incubation time, recovery speed, mortality, and to show the consequences in the simulation result. With such simulation facilities, it is possible to define problems which would be unsolvable in classical training due to the computational burden.

**variability.** When using computers, it is easily possible to offer different versions of similar questions to different students at the same time. This necessitates the student’s trying to solve the problem himself, as other students are given other problems.

**increased teacher availability.** If the standard problems can be handled by a computer program, the human instructors can pay more attention to students with unusual problems. But there is also a need for subject knowledge when building and maintaining the items, so the computer program will also need teacher attention.

**data processing capabilities.** The session results can be processed and used in statistics, e.g. to reveal misunderstandings students might have on a subject, or to show often-made errors.

**cost.** The cost must not be underestimated. It is expensive to make good courseware, since it is usually very specific to one course or even one subject. It is not easy to make generally applicable software.

In conclusion, a computer could provide a tool for increasing teaching effectiveness by providing a more productive learning environment, if precautions are taken.

### 1.2.3 Adaptive computer-based training

If a teacher gives individual training to a student, he or she adapts the level of training to the student’s level of mastery. Training should be of such a level that it is achievable and instructive. “Productive time can be increased by suitting instruction to individual differences and by teaching small-group and individually managed study skills so that students themselves can concentrate more fully on what they require” [Walberg, 1988]. The challenge is to keep it challenging for the student. This is the key issue in adaptive computer-based training (ACBT). The advantage is clear: as the students receive a tailored set of questions, they are challenged but not discouraged, and they are can be productive throughout a complete session.

Of course, there are some problems too. Despite the advantages, not much software has been realised in the field of ACBT to date. There is more literature about computerised adaptive testing or CAT [Schoonman, 1989; Wainer, 1990], but offering training, i.e., trying to improve the student’s proficiency, is a different aim than
assessing the student's proficiency. In control engineering terms, CAT has an observing nature, while ACBT has a more controlling nature. A problem with CAT is the calibration of the items, especially when only a limited amount of data (student answers to items) is available. This problem has to be solved in ACBT as well. Further, the current CAT techniques usually require one single underlying trait, while most engineering courses use more than one trait. This issue is treated in more detail in Chapter 2.

1.2.4 Training in engineering education

The context of the research is a university level engineering environment, in particular, Electrical Engineering. Each year, several hundreds of students start their studies in Electrical Engineering at the Delft University of Technology. Students following this course of study can have specific problems. For example, one cannot see electricity; it can only be measured indirectly; and often complex calculations are required. Many mathematical and physical models are used and much effort is put into simulations and related activities. The ability to interpret the models and simulations in relation to reality is not an easily achieved skill. Most skills to be achieved are problem-solving skills; for example, a design process can often be seen as a problem-solving task.

Consider, for example, a control problem in which the student is asked to examine a motor from which the transfer function is not known. It is possible to apply an input signal (a voltage) to the system and to see the response (the rotation speed). The result of applying a unit step to this system is shown in figure 1.2. The response appears to be zero.

![Figure 1.2. First information to a student on a problem](image)

It is too early to conclude that the motor is defective. The most obvious thing to do is to change the input signal. With a higher step, suddenly a non-zero response appears. Increasing the step size increases the final value of the response until a certain input signal level, after which the response remains constant. The non-linear system behaviour at the beginning can be explained with a dead-zone system. The system is probably a DC motor: you have to put a certain voltage on it before it starts to run, and after reaching its maximum capacity, it cannot go any faster.
This problem shows that the student must solve it by breaking it into pieces, reasoning, choosing, and combining. He or she must be able to make a link between mathematical models and reality. It is not a matter of "insert coin, start calculation and give answer". Training in problem-solving is a good way to acquire the needed skills. However, too often calculation is emphasised at the expense of comprehension issues [Taconis & Ferguson-Hessler, 1994].

A characteristic of many courses is their diversity. They usually consist of a number of related subjects covering a specific research field. Those courses do not necessarily have an increasing level of difficulty, but can follow a path in which prerequisite knowledge of a subject is handled before the subject. A course can even consist of a number of mutually independent subjects.

1.3 The project and its participants

The general objective of the project is:

*Implementation of an interactive system to support training for technical-scientific courses in general, with emphasis on Electrical Engineering.*

Although a lot of research is done in the area of computer-assisted instruction (e.g., intelligent tutoring systems), it is striking to see that very few products of the research are actually used. This project is aimed at a usable system to be applied within the Electrical Engineering curriculum on several courses. Important points of attention are:

- The system must save time for the teachers and instructors. This means that, once running, the system will not need human intervention except for the addition or alteration of items. Further, the system must be capable of generating variants of an item by variation of (teacher-defined) parameters.
- The system must be useful to the students. Important aspects are individual orientation, availability (in time and place) and variability.
- The system must be maintainable. The instructors must be able to implement and alter problems without the assistance of a programmer.
- Since the system is aimed at being generally applicable, it should be as content-independent as possible.
- A lot of specific calculations (transformations, simulations) are made. Taking into account that for many fields software is available for these calculations, it is desired to be able to use these packages, either off-line (e.g., calculating in advance and just using the results) or on-line (e.g., calculating when needed).
Introduction

- In many courses, diagrams, graphs, etc. are used intensively. As these graphics play an important role in subject comprehension [Honderd & Jongkind, 1990], the system must be able to show such graphs to the student.

The project, funded by the Onderwijs Stimulerings Fonds (Education Stimulation Fund) of the Delft University of Technology, started as a cooperative effort between the Control Laboratory of the Department of Electrical Engineering, and the Educational Development Unit of the Department of Philosophy and Humanities, both of the Delft University of Technology. Soon it appeared that various laboratories at the university were interested in such a system: a new project named CATE (Computer Assisted Teaching in Electronics) was initiated by the laboratory of Electronics. It was decided to join forces and the scope was broadened to the Department of Electrical Engineering, with, as two pilot courses, Control Engineering and Electronics.

1.4 Synopsis of the thesis

The thesis describes the development and introduction of a computer-assisted training system to be used in a real environment.

The outline of the thesis is as follows: In Chapter 2, an overview of computer support in instruction is given, with the aim of training in mind. Chapter 3 focuses on some artificial intelligence techniques which could be useful in the project. Chapters 4 and 5 describe the design of the system, especially the adaptive part of the management system. Chapter 6 gives an overview of demands on the item implementation environments. Further it contains a description of the chosen environment Taiga, and the design of Taiga extensions for interfacing with the management system, and for support of Control Theory related graphs. The system has been tested by students; Chapter 7 describes these tests and the results. Finally, in Chapter 8, conclusions and suggestions are stated. The dependencies are shown in figure 1.3.

![Figure 1.3. Structure of the thesis](image)
Computer support in training and testing

This chapter starts with a description of the history, structures and control strategies for computer-assisted instruction (CAI). Then several methods are described which could possibly be used for training systems: item banking, adaptive strategies in testing, and intelligent tutoring systems. The conclusion describes the usability of these techniques and several suggestions concerning the training system design.

2.1 Computer-assisted instruction

2.1.1 History

Even before the first computers, people were trying to build machines to assist instruction. In 1926, Pressey made a mechanical learning device. It was able to ask questions; the students could answer in a multiple choice format. If the answer was correct, the student was rewarded with a piece of candy.

The next step was the Behaviourist theory of Skinner around 1958 (see figure 2.1).

![Figure 2.1. Skinner's model](image.png)
Programmed Instruction (PI) evolved, followed by the first CAI, implemented around 1963. Crowder refined the linear PI model and implemented an "auto tutor" using a branching strategy to offer alternative routes through the instruction. In 1967, the Computer Curriculum Corporation was formed after research at Stanford University [Suppes et al., 1968], and in 1968 the PLATO project, a cooperative project between the University of Illinois and Control Data Corporation, produced results. Until the late seventies, the majority of CAI programs concerned drill & practice, on large centralised computers. In the early eighties, microcomputers began to appear in schools, mostly via the initiative of enthusiastic teachers. Authoring languages started to appear, designed to enable non-programmers to create frame-based CAI lessons. Unfortunately, teachers often found that it took more time to build courses than they had available.

The main problem with CAI is that it depends heavily on what the software designer has anticipated, although quite sophisticated programs were built with branching strategies to provide some adaptation. However, the adaptation was rigid (coded into the program). Therefore many research activities were started to make more intelligent CAI programs. Although some had already started in the seventies, the year 1982 is often seen as the starting year of intelligent tutoring systems or ITS, with the publication of the book "Intelligent Tutoring Systems" [Sleeman & Brown, 1982]. Another direction followed is the adaptive CAI research. Both are discussed below.

2.1.2 Structures

CAI programs can contain one or more structures in order to help students meet their objectives. Godfrey & Sterling discriminated five basic teaching strategies [Godfrey & Sterling, 1982]: Drill and practice programs provide practice in a circumscribed curriculum domain, which assumes that the student has previously been taught the content and now requires practice with feedback. Many of the early CAI applications fall into this category. Drill and practice is especially useful for teaching in simple subjects for large numbers of students. Tests assess the student's level and diagnose possible deficiencies, giving a judgement about the student's performance. With inquiry, the student possesses the initiative to choose rules and examples, much like using an encyclopedia. A suitable implementation is a hypertext environment, in which the student can browse around. In simulations, dynamic representations of complex domains (e.g., economics, industrial processes, aeroplanes) are created. This allows students to do all kinds of experiments without influencing real processes. It is especially useful if the actual process is very expensive to operate, or if it is potentially dangerous to make mistakes. Simulations encourage inductive
learning. *Tutorials* combine rules, examples and questions to teach a skill or subject and to provide practice. Especially over the last decade, research has been performed into intelligent tutoring systems. Characteristic of those systems is the diagnostic function to determine possible causes of repeated errors, misconceptions, etc. Good intelligent tutorials are very difficult to build, mainly because of the adaptivity factor.

### 2.1.3 Control strategies

**Student control**

After the first generation CAI systems in which the student had almost no control, later systems tended to put the student in full control. Especially in intelligent CAI, student control is a common control strategy, in which the student can direct the flow of instruction provided by the system. This strategy relies on the assumption that the student is able to decide for himself or herself what is needed. There are several degrees of learner control. In a limited form, the student can only control the pace of instruction. More student control is offered if it is also possible to influence the content (selection of lesson, objective or item, e.g. by means of a menu), or strategy (presentation forms, cognitive processing). The degree of student control is large if the student can define his/her own problems, for example, in microworlds, where a simulation world is created in which the student can explore the possibilities.

**Program control**

Complementary to student control is program control, where the program makes the decisions. In a non-adaptive form, all students receive the same amount and type of instruction. Early CAI systems are examples of system with a high degree of non-adaptive program control. In adaptive program control, the instructional strategy to be used is based upon performance or individual differences, such as amount, type, sequence, reading and answering time, context/layout and feedback.

**Discussion**

The choice between program control and student control has been discussed intensively. The results of studies are sometimes conflicting: some studies report an increase in learning with student control, others report a decrease. Student control usually leads to a more positive student attitude [ten Bruggencate et al., 1993].
However, especially with low achievers, student control may be ineffective. Low achievers are likely to need more guidelines [Chung & Reigeluth, 1992].

A test with secondary technical school students [Hasselerharm & Leemkuil, 1990] showed no significant differences between student control, program control and adaptive program control for post-test scores. However, adaptive program control was found the most efficient, i.e. the same performance level is reached in a shorter period. Although the test was done with secondary technical school students, a broader validity of this outcome is probable: as the students receive more instruction on their level, less time is wasted, and therefore a shorter period should be needed to reach the desired level.

A compromise between the freedom with student control and the guidance with program control would be student advisory control: suggest an action to do next, but leave the decision to the student. In a somewhat more program-controlled form, the student could be given a list of possible actions, of which one can be chosen.

2.2 Item banking

Item banking systems are often used in non-adaptive computer-based testing. The basis of such a system is an item bank consisting of a large number of problems together with their characteristics. Each unit is called an item. An example of an item banking system is the interactive test and exercise management system Items-D [Jansen et al., 1991]. The structure of Items-D is shown in figure 2.2. It

![Diagram of Item Banking System](image)
2.2 Item banking

consists of two programs, the development system which allows the user to define tests, implement items, and keep a registration of students, and the distribution system providing the tests to the students. The distribution system is student controlled in the sense that the student can choose the order in which to answer the items, the student can suspend the answering of an item and resume later if desired, and the student can use his/her own pace. It is program controlled in the sense that the items to be given cannot be influenced by the student.

Unlike most other item banking systems, Items-D does not have its own language for the implementation of items, but it relies on external programs for that. The advantage of this choice is that the teacher can use the implementation environment most suitable for an item. Simple static items can be made with an authoring system, for example, and items incorporating simulations can be made with a programming language or a special system. Naturally, for the distribution system to be able to run the items, the item implementation environment must allow an item to be run stand-alone.

Figure 2.3 displays the strategy used by Items-D to define a session. The teacher defines tests with a profile, containing criteria for selection of each item in the test.

![Figure 2.3. Problem selection scheme with profiles](image)

This method makes it possible to create several parallel versions of a test session. The versions are created in advance. Items-D also supports variation of parameters and answers. Each item contains a description of acceptable values or calculation methods for parameters and answers. When a test is instantiated (i.e., when a concrete test is built from the profile) these variables are assigned values as well.
The strategy also shows a limitation of Items-D. It is non-adaptive; sessions are completely specified beforehand. Although individual students usually get different sessions than their neighbours because there can be several parallel sessions, the sessions do not adapt to the individuals. A computer can do better than that if all problem characteristics are available. Further, there is still much teacher dependency, since the teacher must devise the sessions from the profiles, and change them now and then to prevent the items from "becoming known". In a training environment, teacher dependency is undesirable. And because the number of different sessions is predefined and therefore limited, there is still a reasonable probability of two students receiving the same session. Several adaptive testing strategies have been developed; the next section describes the most popular of these.

2.3 Adaptive strategies in testing

2.3.1 The link between training and testing

In a report on decision-making for adaptive instruction [van Bruggen & van de Ven, 1989], an overview of research into learner characteristics and adaptation is given; it argues that intelligence and prior knowledge are the strongest predictors of learning performance. This makes them the most likely bases for adaptation in training. Instruction should be "challenging but not frustrating" [Brophy & Alleman, 1991]. However, intelligence and prior knowledge are unobservable (and multidimensional) entities; they can only be "measured" partly and indirectly, by observing the student's reactions.

A general approach to adaptive training has not been found in literature. More literature is available on adaptive testing. In both cases, a decision about mastery is made: in testing to give a judgement about the mastery level, in training to select another problem, which hopefully assists in improving the mastery level. The similarities are close enough to justify a closer look at adaptive testing strategies.

Adaptive testing methods use the analysis results of answers to previous questions in order to adapt to the students. Several methods of adaptive mastery testing are discussed in this section. First, two methods of adapting the test length are described, then a method of adapting both the test length and item difficulty. With these three methods, it is assumed that the items measure the same underlying skill or subject mastery level. In other words, it is assumed that the item pool is unidimensional. A multidimensional method of adapting the skill or subject being tested, the knowledge unit adaptation, is discussed next. The section concludes with a discussion about the usability of the methods.
2.3.2 Test length adaptation

Sequential probability ratio test

The Sequential Probability Ratio Test (SPRT) was published by Abraham Wald just after the Second World War [Wald, 1947]. It is a simple but elegant method, based on the well-known binomial model, in which information is gathered in order to choose between two alternatives. It has been widely used for quality control in manufacturing, but it is also applicable in computer-based testing. In this case, the alternatives are usually mastery and nonmastery. The information on which the decision is based is the observed sequence of a student’s correct and incorrect answers to test items. It is assumed that the observations are dichotomous and a that random sampling without replacement occurs.

The method uses the following algorithm [Frick, 1990]:

After each observation, a probability ratio (PR) is determined according to Eq. 2.1, in which \( s \) is the number of observed successes, \( f \) is the number of failures, \( P_m \) is the probability of selecting an item that a master answers correctly, and \( P_n \) is the probability of selecting an item that a nonmaster answers correctly.

\[
PR = \frac{P_m^s (1 - P_m)^f}{P_n^s (1 - P_n)^f}
\]  

(2.1)

Using the error rates \( \alpha \) (the probability of misclassifying a nonmaster as a master) and \( \beta \) (the probability of misclassifying a master as a nonmaster) a decision scheme can be constructed. The structure of an SPRT is shown in figure 2.4.

![Figure 2.4. Structure of the SPRT model](image_url)
The number of items needed to make a decision depends on the student’s performance. The test length is influenced by the error rates (higher values will yield shorter tests, but more errors) and by the gap between \( P_n \) and \( P_m \) (the larger the gap, the shorter the average test). These last two values can be updated empirically by using average proportions-correct for both groups.

For example, assume \( \alpha = \beta = 0.05 \), \( P_n = 0.4 \) and \( P_m = 0.9 \). If a student answers 4 items all correctly, mastery will be assumed. If the student fails on one item, there will have to be 6 correct answers before the algorithm decides on mastery.

The advantages of this method are that it is very easy to implement and it does not need parameter estimations (e.g., difficulty) for each item. SPRT has been criticised because it is possible that, by chance, students will get very easy or very hard questions right at the beginning, thus causing the algorithm to decide prematurely on mastery or nonmastery. According to Frick, to prevent this, the items chosen will have to be representative and the error rates should be chosen to be small enough [Frick, 1989].

An assumption of SPRT is independence. This means that the probabilities must not change according to which items have been presented. If SPRT is used in a training environment, this requirement is easily violated, for example, when giving feedback.

**Bayesian Posterior Beta Probabilities**

Another method described by Frick is the use of Bayesian posterior beta distributions. This distribution is used in order to calculate the probability that \( \Phi \), an estimate of the student’s true proportion-correct, is greater than a specified criterion (or cut-off) score, given an observed number of successes \( s \) and failures \( f \). This method was used by Tennyson in the MAIS system [Tennyson & Christensen, 1988]. The probability density function for the posterior beta distribution is

\[
\beta(\Phi|s,f) = \frac{(s+f+1)!}{s!f!} \Phi^s (1 - \Phi)^f
\]  

(2.2)

For the probability distribution function, needed to calculate the probability that the true proportion-correct exceeds a certain value, a numerical integration method is needed. Then, analogous to the SPRT, a probability ratio can be defined:

\[
PR = \frac{P(\Phi \geq \Phi_e|s,f)}{P(\Phi < \Phi_e|s,f)}
\]  

(2.3)
with which the same method of decision can be used. As the method is very similar to the SPRT, the same advantages and criticisms apply. The main difference between this method and the SPRT is that it utilises a single cut-off for mastery decisions. A disadvantage is the numerical integration, but with the current computing power of even low cost personal computers this is not very significant.

2.3.3 Test length and level adaptation: Item Response Theory

A consideration not addressed in the previous methods is that although all items in the pool may measure the same trait, the information they provide is not identical. In Item Response Theory (IRT) this is formalised by explicitly assuming one single dimension of knowledge or underlying trait for all items.

The models

A first observation is that items usually do not have the same difficulty. In IRT, the item difficulty, denoted \( b \), is defined as the position that the item occupies on the knowledge dimension. The position of the student on the dimension is called mastery level or proficiency, denoted \( \theta \). IRT models give the probability that a student will answer a question correctly in terms of these quantities. The simplest model, developed by the Danish mathematician Georg Rasch, is called the one-parameter logistic model or 1-PL [Rasch, 1960]. This one parameter is the item difficulty:

\[
P(\theta) = \frac{1}{1 + e^{-(\theta - b)}}
\]

(2.4)

In words, \( P(\theta) \) gives the probability that someone with proficiency \( \theta \) will give a correct answer to an item with difficulty \( b \). Graphical representations of this probability, as shown in figure 2.5, are called Item Characteristic Curves (ICC).

Another important characteristic of an item is the item discrimination. This is often interpreted as a measure of quality for the item; the higher the value of the item discrimination is, the better it distinguishes between students with a lower proficiency than the item difficulty and those with a higher proficiency. Including the item discrimination, denoted \( a \), in the model yields the two-parameter logistic model or 2-PL:

\[
P(\theta) = \frac{1}{1 + e^{-a(\theta - b)}}
\]

(2.5)
The discrimination defines the slope of the ICC. Steeper slopes imply higher values for $a$ as shown in figure 2.6.

The 2-PL model is a great extension to the original model. Data can be fitted much better using the extra parameter, but there still remains one parameter which should
not be ignored. A non-negligible influence in the chance of correct response is caused by the very often used multiple choice items. Students have a substantial chance of guessing the correct answer. For example, with a 4-choice item, the probability of guessing correct is 0.25. This guessing parameter, denoted $c$, is included in the three-parameter logistic model or 3-PL:

$$P(\theta) = c + \frac{1 - c}{1 + e^{-a(\theta - b)}}$$

(2.6)

This parameter has the effect of starting the curve at a non-zero probability as shown in figure 2.7.

![Graph showing the influence of $c$ in the 3-PL model](image)

*Figure 2.7. Influence of $c$ in the 3-PL model ($b = 0$, $a = 1$)*

**Estimation of proficiency**

Assuming the parameter vectors $\beta_j = (a_j, b_j, c_j)$ are available for all items in the pool, the student’s proficiency $\theta$, given a set of items responses $x$, can be estimated. For this, the method of *maximum likelihood* is used. The likelihood function $L$ gives the likelihood that a certain response pattern would be observed if $\theta$ were the actual value (see Eq. 2.7).

$$L(\theta | x, \beta) = \prod_j P_j(\theta)^{x_j} (1 - P_j(\theta))^{1-x_j}$$

(2.7)
In words, it is the multiplication of the probabilities that each item response will occur. If an item was answered correctly, the ICC is taken, else the complement. For example, assume a student answered two items. The first, with difficulty 0, was correct, and the second, with difficulty 1, was incorrect. In figure 2.8 the process is displayed graphically. The maximum of this likelihood function gives the most likely value for the proficiency.

![Graph showing probability against proficiency](image)

**Figure 2.8. Construction of response likelihood distribution**

Now that it is possible to find an estimate, we want to know how accurate the estimate has become. The error variance ($\sigma^2$) indicates the width of the likelihood function. A popular method to estimate that variance uses the information function:

$$I(\theta) = -E \left\{ \frac{\partial^2}{\partial \theta^2} \ln L (x| \theta) \right\} = \sum_i \frac{\left( \frac{\partial}{\partial \theta} P_i(\theta) \right)^2}{P_i(\theta) (1 - P_i(\theta))}$$  \hspace{1cm} (2.8)

in which $E$ is the expectation. Maximum likelihood estimators are asymptotically Gaussian [Candy, 1986]. It can be deduced that the information function for a Gaussian distribution $N(\mu, \sigma^2)$ is equal to $1/\sigma^2$. Because the estimator property is asymptotic, using the information function is less accurate for small numbers of items. However, as the information function is additive over items and does not depend on the response pattern, it is a reasonable choice for practical work.
If the administered items are appropriately difficult, longer tests yield narrower curves. By defining a criterion value for the reciprocal of the information at level \( \theta \), a stopping rule for the process is given by a certain accuracy in the measurement.

A key issue is the selection of appropriately difficult items. Items that are too easy or too hard do not provide any information, because the ICCs of such items are horizontal in the region of interest. Thus, an appropriately difficult item is an item that provides much information. Therefore, the information function can be used for the selection process as well. Since the information function is additive over items, and it does not depend upon the item responses, it is possible to calculate the influence of any item and select the item giving the most information.

The method described above does not use any prior information about the students. This can be taken into account by introducing a prior distribution and using the Bayes Modal Estimate. Further problems exist, such as the estimation of item parameters. Given enough item response vectors, the item parameters can be calibrated. However, enough is usually quite a large number. These issues are beyond the scope of this thesis. For the interested reader, Item Response Theory is handled in more detail in a good primer [Wainer, 1990].

Frick describes another method to estimate the proficiency [Frick, 1990] which was developed by Owen, called the Bayesian posterior estimation. With a prior Gaussian distribution and the item result, this method derives a mean and a variance for a posterior distribution. The difference between this method and the maximum likelihood estimator is that, for every estimation, the likelihood function is transformed into a Gaussian distribution, while the maximum likelihood estimator uses the real function. An advantage is that much less memory is needed to store the function, and computations are easier because the shape of the curve is already known; a disadvantage is that information is lost.

As was to be expected, especially in the first few items of a session, the results using this method can be quite different from the maximum likelihood estimation. After a few items, the estimates by the two methods tend to converge.

**Score assignment**

A specific problem is the assignment of scores. The final proficiency estimate, a real valued number between minus infinity and plus infinity, is not a common method to present the student's ability. The raw score, i.e. the total number of correct answers, is no reliable entity, because students get items with different characteristics. In the ideal case, half of the items will be correct and half will be incorrect. A frequently used method is making use of the item pool score, given in Eq. 2.9.
\[ e(IPS) = \sum_j P_j(\theta) \]  \hspace{1cm} (2.9)

This can be interpreted as the expected score of a student with a certain proficiency, if given all items in the pool. The item pool score can also be used for a stopping rule which is more compatible with the previous methods of cut-off scores. In this rule, the item pool score is used to define a cut-off proficiency \( \theta_c \). The process stops if the confidence interval for \( \theta \) no longer contains \( \theta_c \). It is safe to conclude mastery if the interval is completely above the cut-off proficiency, and nonmastery can be assumed if the interval is completely below the cut-off frequency.

**Extension to allow multiple attempts**

The IRT method as just described does not take into account that it can be allowed to let the student answer more than once to an item. In a testing environment, this is usually not desirable, since giving hints as to whether or not the answer was correct can influence the proficiency. However, in a training situation, it is not unusual to give a hint if an answer was incorrect, and let the student try again. Clearly, a student who gives a correct answer immediately is likely to have a higher proficiency than a student who needs a second try. Therefore the likelihood estimation is extended. The likelihood that a student with proficiency \( \theta \) will answer item \( i \) correctly in \( n \) attempts is

\[ L_i(\theta) = (1 - P_i(\theta))^{n-1} P_i(\theta) \]  \hspace{1cm} (2.10)

and the likelihood of answering incorrectly all \( n \) attempts is

\[ L_i(\theta) = (1 - P_i(\theta))^n \]  \hspace{1cm} (2.11)

Comparing this definition with the earlier likelihood function for single attempt items, the difference is the answer vector \( x \), which had values 0 (incorrect) or 1 (correct). Combining equations 2.10 and 2.11 into one equation using \( x \), the likelihood that a student shows a certain response pattern \( (x, n) \) becomes:

\[ L(\theta|x, n) = \prod_i P_i(\theta)^{x_i} (1 - P_i(\theta))^{n_i-x_i} \]  \hspace{1cm} (2.12)

Thus the incorrect attempts before a correct answer is given are also considered. However, this modification does not take into account the influence of the guessing
2.3 Adaptive strategies in testing

parameter. With a multiple choice format, the odds are very small that a student will choose the same alternative in the next attempt. It is more likely that one of the remaining alternatives will be chosen. For example, with a 6 choice item, there is a 1/6 probability of guessing correctly, but after an incorrect answer one alternative is ruled out, leaving a probability of 1/5. This increased probability could be modelled by making the guessing parameter $c$ dependent on the number of attempts used (assuming the number of alternatives is larger than or equal to the maximal number of attempts):

$$c_n = \frac{c_{n-1}}{1 - c_{n-1}}$$  \hspace{1cm} (2.13)

in which case the probability of answering item $i$ correctly the $n^{th}$ time becomes

$$P_{i,n}^{(\theta)} = \frac{1}{1 - (n-1)c} \left( c + \frac{1 - nc}{1 + e^{-a(\theta - b)}} \right)$$  \hspace{1cm} (2.14)

The likelihood of occurrence of a response $(x, n)$ to an item $i$ becomes:

$$L_i(\theta|x, n) = P_{i,n}^{(\theta)} \prod_{j=1}^{n-x} (1 - P_{i,j}^{(\theta)})$$  \hspace{1cm} (2.15)

For the sake of clarity, if $n = x = 1$, the upper limit of the product is lower than the lower limit; in that case the product operator leaves the argument unchanged.

This guessing parameter correction is not generally applicable, as with non-multiple choice questions the probability of guessing correct will usually not increase equally fast. It is even imaginable that the guessing probability will not increase at all.

2.3.4 Knowledge unit adaptation

The previous methods all assume the unidimensionality of the underlying trait to be tested. In practice this is often not the case. Many courses consist of several skills, subjects, or concepts to be mastered. So another level of adaptivity can be applied, the decision what to test the student on. An adaptation method acknowledging this is the concept lattice method. This method uses a model of the relations between knowledge units (e.g., skills, concepts or subjects) to be tested, represented via an and-or graph [Doignon & Falmagne, 1985; Kohnert & Janke, 1990]. Knowledge units are represented by vertices, and precedence- or logical links between them are
represented by edges. And-edges between a vertex and its children denote that all child vertices must be mastered before the vertex can be mastered, or-edges denote that mastery of one of the child vertices is enough. An example of an and-or graph is shown in figure 2.9. This figure shows that to master f, both d and e are needed; to master d, both a and b are needed and to master e, either b or c is needed. With the knowledge structured in a lattice, it is possible to test the students efficiently.

Figure 2.9. Example of an and-or graph

Eskenasi suggests a method that is particularly useful for multiple choice items [Eskenasi et al., 1993]. All answers to items are associated with a vector, the size of which is equal to the number of knowledge units:

\[ \underline{a} = (a_1, a_2, \ldots, a_n) \quad (2.16) \]

The elements of the vector indicate if the corresponding knowledge unit participates in obtaining the answer. The elements can have values 1 (if the knowledge unit must be used correctly to obtain the answer), 0 (answer does not imply anything regarding knowledge unit) or -1 (answer indicates lack of knowledge of the skill). The value -1 occurs only with incorrect answers. The student's knowledge is modelled by a vector

\[ \underline{b} = (b_1, b_2, \ldots, b_n) \quad (2.17) \]

in which the elements can take the values 1 (if the student is aware of the skill), 0 (if the skill is not tested) or -1 (if the skill is not known).

Eskenasi suggests two adaptation strategies for skill selection:

**Full testing.** All prerequisites of a goal are tested even if they are connected with an or-link. At the end, the student model will contain full information about the degree of knowledge.
2.3 Adaptive strategies in testing

**Quick testing.** All prerequisites are tested only if there is an and-link between them. This strategy can be used when the goal is to assess if the student has the minimum knowledge needed to accomplish the goal.

If it is decided which skill will be tested, an item will have to be selected. First, the item set is restricted to those items for which the corresponding skill element has the value 1. For further selection, again two strategies are suggested:

**informative:** choose an item $k$ maximising the difference between item vector and student model:

$$k \left| \sum_i |b_{ki} - a_{ki}| - \sum_i |b_{ji} - a_{ji}|, j \neq k \right.$$  \hspace{1cm} (2.18)

**adaptive:** choose an item corresponding to knowledge. The item weight, defined as the number of skills occurring in the correct answer, scaled on some interval (1, w), is used to describe item difficulty. A student "weight" (proficiency) is calculated by summing the vector elements of the student model and scaling the result on the same interval. The item weight could be fine tuned with a correction factor, initially supplied by the item designer and possibly tuned by means of statistical analyses. This provides a flexible implementation of a course model. It is possible to build a subject tree, specifying the prerequisites for each subject. It is also possible to specify the subjects for each problem. A subject account vector can reveal that a student has not mastered a particular subject. Via the subject lattice, possible causes of the deficiency can be found and a matching problem can be fetched.

### 2.3.5 Item form adaptation

Items can have several forms. Some students may prefer graphics, others texts. Eskenasi suggests a p-vector associated with each form of an item [Eskenasi et al., 1993]. This vector contains four parameters: the levels of intelligence, concentration, and self-confidence, with values -1 (low), 0 (medium) and 1 (high), and the psychopedagogical (PP) type of the question, coded as an integer. It represents the characteristics of a student for whom the form would be the most appropriate. For the student, a corresponding vector, called the m-vector, is maintained. The student's characteristics are used to select an appropriate form, and the student's ability to handle the selected items gives an impression of his characteristics. Again, there are two strategies: to minimise the difference between the three m-values and p-values of the form, or to select a form corresponding to the most successful PP type for the student.
2.3.6 Discussion

Several methods of adapting tests to the students have been described. To what extent is it possible to use one or more of these methods in an adaptive training system for engineering courses?

A multidimensional approach is needed when the aim is support for a multiple subject course. In the concept lattice method, the difficulty of items is completely specified by the number of participating concepts. It is reasonable to assume that, in practice, the lattice will not be specified in a way that is detailed enough for such an approach. Some concepts will need general or prior knowledge not explicitly defined. Further, it is not unlikely that two items using the same concepts will still have different difficulties. So the problems will contain difficulty variations unexplainable by the lattice.

The length adaptations and the length and level adaptation are unidimensional and therefore do not seem appropriate if used alone. However, combination of the concept lattice method and a unidimensional method could solve the dilemma. The course is then divided into a number of subjects which can be represented in a subject lattice. If properly chosen, each subject can be considered unidimensional; the proficiency of the student in those subjects can be measured with one of the unidimensional methods. When generalising the method by including different item formats, there are three levels on which a system can adapt:

- the course level, by selecting a suitable subject,
- the subject level, by selecting a suitable item,
- the item level, by selecting a suitable format.

Depending on the number of participating students, available data, and teacher time, there are several alternatives for the subject level adaptation. The more sophisticated IRT method or the more simple Beta or SPRT method can be used. A problem could be formed by the assumption of the methods that the student’s mastery level does not change, as the methods were designed for testing. With training, the goal is improvement of mastery level. Modifications to overcome this problem may be needed. For example, the IRT method of item selection might not be appropriate. The most informative item is not necessarily didactically the best item: in training, usually a gradual increase in difficulty is preferred [Brophy & Alleman, 1991], as opposed to the likely possibility that IRT will choose the most difficult item after the first, simple, item has been answered correctly.
Intelligent tutoring systems are often designed for training tasks. Therefore, the basics of those systems are described next, to investigate possible usability.

2.4 Intelligent tutoring systems

Intelligent tutoring systems (ITS) involve the application of artificial intelligence techniques to instructional programs [Sleeman & Brown, 1982; Wenger, 1987; Costa, 1992]. The research area is also called intelligent CAI or ICAI. ICAI programs constrain themselves to a specific portion of a domain. Nearly all ITSs are coaches, meant for expanding and correcting the application of knowledge in a specific domain [Winkels & Breuker, 1992]. They do not present the student with a predetermined body of facts to be learned, but rather serve as a problem-solving monitor, coach, or consultant.

Intelligent tutoring systems can be seen as a special type of expert system, with a domain expert having expertise on the subject matter, a student expert modelling the knowledge and/or problems of the student and a didactic expert knowing how knowledge should be transferred (see figure 2.10). Sometimes a fourth expert, the interaction expert, is distinguished, but this can also be seen as a task for the didactic expert.

![Diagram of ITS architecture]

Figure 2.10. Traditional ITS architecture

Lesgold et al. categorise required knowledge as domain expertise (expert procedures and their expected results), curriculum knowledge (specification of the goal structure guiding teaching), instructional and test planning knowledge (metacurricular knowledge such as extraneous sources of difficulty), and treatment knowledge (information on problem formats and feedback) [Lesgold et al., 1989].

Emulating human teachers has proved to be very difficult. Human tutoring is inconsistent, content and context sensitive, and relies on natural language and nonverbal signals [Kurland & Kurland, 1987]. These are all characteristics which are very hard and sometimes even impossible to incorporate in an ITS. The bandwidth problem,
Computer support in training and testing

i.e. the fact that a computer is not capable of modelling the student completely due to the limited communication, has become very clear during ITS research. Therefore another trend is to use less detailed models to classify the students globally in a number of subgroups; such systems appear to be quite useful in practice.

A more detailed structure of a generic intelligent tutoring system for problem solving activities is shown in figure 2.11 [Kyllonen & Shute, 1989].

![Diagram of an intelligent tutoring system](image)

*Figure 2.11. Components of a generic intelligent tutoring system*

The original ITS model shown in figure 2.10 suggests independence of the experts. However, there is an overlap in knowledge needed for the experts, and also the functions of the experts overlap [Winkels & Breuker, 1992]. For example, the domain expert and the student expert both use the same knowledge base when examining whether the student made an error and what caused the error. Winkels & Breuker suggest that the possible cause of confusion is a too anthropomorphic view
of the functions [ibid.], and present a metaphor that sees coaching as three-level process control, with as functions for all three levels planning, monitoring and diagnosing. The three levels are the curriculum level, the task level and the communication level. In the curriculum level the educational goals are planned, the results are evaluated and possible adjustments are done. The task level embodies control of the actual problem-solving process. Finally, the communication level uses a discourse plan to transform speech acts into expressions.

The process control model suggests a more distributed approach to tutoring. There is a clear relation with the model described in section 2.3.6 (page 26). When fitting, that model appears to be on the curriculum level in the process control model; the task and communication levels are within the items.

2.5 Conclusion

This chapter has described several approaches to CAI programs as presented in literature. For a general training system, traditional CAI is too rigid. Adaptations are limited to branching strategies and the programs are content dependent. These structures are not flexible enough to adapt to individual differences; other solutions will have to be found.

Two usable control strategies are student control and adaptive program control. The advantages of both methods can be obtained by using student advisory control, in which the program gives advice, based on an adaptive mechanism, but the student is in control.

The use of an item banking system is a possibility for the training system. This approach is generally applicable, but current item banking systems are not adaptive to the students, or restrict the implementation of items to a certain environment. A possible combination of an open item banking system such as Items-D and an adaptive strategy for test and training session generation seems a promising option.

For adaptation, a hybrid strategy, in which a subject lattice is used for high level (subject or skill) selection, and a unidimensional method for lower level (difficulty within the subject or skill) selection, seems an almost natural choice. As a possible third level, a format selection could be introduced, based on student characteristics.

It is open to discussion if the IRT item selection method is appropriate in training. Often a gradual increase in difficulty is desired, which makes the IRT method inappropriate. However, in other circumstances the IRT method may be better. Therefore, the system must be flexible enough to allow different strategies.
Due to their specific nature, a single ITS is also not a suitable architecture for the system. However, it is imaginable that an ITS can be incorporated for specific subjects, as a special kind of item. Further, the results of ITS research can be applied in the system. Because the system is aimed at training for courses, with dependency between subjects, a curriculum model and a student model will be needed, to be able to make reasonable decisions about the next item to be presented. The process control metaphor seems an appropriate basis.

In conclusion, this chapter has shown a direction in which to proceed in designing the training system. It is suggested to make a system which takes into account that courses consist of often incomparable subjects. In other words, a multidimensional approach is advocated, with a two-level or possibly a three-level adaptation mechanism. The domain knowledge, consisting of items, will remain the teachers’ responsibility, outside the system. This choice was made for two reasons: many teachers have items already available, and separate domain knowledge makes a system more generally usable.
Artificial intelligence and computer-based training

3.1 Introduction

In Chapter 2 it was argued that it is not unreasonable to look at artificial intelligence (AI) techniques when designing a program for instructional purposes. This is supported by an early definition of AI by Minsky:

AI is the science of making machines do things that require intelligence if done by men [Minsky, 1962].

This chapter describes some typical AI methods which could prove useful in instructional programs. The first are expert systems. Limitations to the first generation of expert systems have initiated research into a new generation of expert systems and other methods such as machine learning. As an additional feature, these techniques are often based on a model of human thinking; this could be usable in educational purposes as well. Further, distributed systems have been developed, of which the blackboard architecture is described in more detail. The descriptions are accompanied by considerations concerning their application possibilities in a training system.

3.2 Expert systems

Over the last decades, much has been said and written about expert systems. To start with, two characterisations of expert systems are given.
An expert system is a specialist system embodying the expertise of a very narrow domain [Keravnou, 1990].

An expert system is a knowledge-intensive symbolic computer program that solves problems of a limited domain normally requiring human expertise [Humpert, 1990].

The types of expert systems currently present are described in the following sections.

### 3.2.1 First generation expert systems

A generic framework for a first generation expert system is given in figure 3.1. It consists of a knowledge base, containing domain information, and an inference engine reasoning with the knowledge. Often, a separate data base is distinguished,

![Figure 3.1. First generation expert system architecture](image)

the knowledge base containing general structures and the data base containing facts. The most common architecture for first generation expert systems is the rule-based architecture. In a rule-based system, the knowledge base consists of a set of production rules and a data base of domain information. A production rule consists of two parts, the antecedent part with the conditions to fulfil and the consequent part stating the conclusion to be drawn or the actions to perform:

IF <conditions> THEN <actions>

Actions can be stating or changing variables, executing external functions, or presenting information to the user. If a variable is set, it becomes a fact. A set of facts, the domain information, forms a description of the problem domain.

First generation expert systems use a deductive line of reasoning. Two inference mechanisms can be distinguished: forward chaining, in which conclusions are
drawn on basis of "what is available" and backward chaining, which works on basis of "what is needed". In other words, forward chaining starts at the antecedents and backward chaining starts at the conclusion. For example, consider a very simple knowledge base:

data base (facts):
   A
rule base:
   1: IF A THEN B
   2: IF B THEN C

goal:
   C?

With forward chaining, we start at the antecedents. We know the fact A so we search for rules with A in the antecedent. Rule 1 applies, so we can infer a new fact B. This fact allows the use of rule 2 to infer C. This was the goal, so we stop. Backward chaining works the other way around. We want to know C. So we search for rules with C as conclusion. Rule 2 applies, so if we can prove B, C can also be proved. Rule 1 has B as conclusion. So if we can prove A, B can be proved. A is a fact. Nothing remains to be proved, so C can be concluded.

The forward chaining process has worked well for selection, diagnosis and consultation applications, or, more generally, problems for which the number of possible solutions is large. The backward chaining is suitable for solving goal oriented problems requiring broad, but shallow, knowledge [Humpert, 1990].

Intelligent tutoring systems often use rule-based methods to enter domain knowledge for problem solving, to update student models and to make tutoring decisions.

Limitations

Although rule-based methods can be very useful (examples can be found in the oil industry and in finance), there are severe limitations, which can be divided into three classes [Keravnou, 1990]:

human-computer interaction. Question series are often incoherent, redundant questions may appear, no case history is maintained, user interfaces are inflexible, and users are not allowed to revoke an answer or pursue the effects of an alternative answer, and explanations are often not meaningful or not meaningful enough.

problem-solving flexibility. Performance degrades dramatically when dealing with difficult (rare) cases, it is unable to recognise that a problem case is at the periphery
Artificial intelligence and computer-based training

or outside its area of expertise, and the user is not allowed to volunteer information or focusing guidance.

extensibility and maintainability. It is difficult to modify the system knowledge, both manually and automatically, consistency checks are not facilitated and the system is not able to evolve on the basis of its experiences, i.e. it does not learn.

Many of the shortcomings can be contributed to the shallowness of the reasoning process. Characteristics of shallow reasoning are the use of rules of thumb and heuristics. In rule-based reasoning, the model of the expertise is not explicit. To clarify this, Keravnou gives as an example a rule of the well-known Mycin system. The specific information is omitted to focus on the real problem:

if       <specific information>
and      the patient is at least 17 years old
and      the patient is an alcoholic
then     there is suggestive evidence for <specific information>

The reason to insert the age condition is to prohibit Mycin from asking if a child is an alcoholic. A side effect of this rule is that the age of the patient is always needed, although it is only important to know whether the patient is an alcoholic. The problem is that the association of alcoholism and age is general knowledge, as opposed to specific expertise linking alcoholism to a cause of the disease in question.

3.2.2 Second generation expert systems

Many problems with first generation expert systems are caused by their shallowness. Therefore, the goal of second generation expert systems is to use more deep knowledge, i.e., reasoning from domain first principles and domain structure. For this, knowledge must be modelled task independently. Since domain first principles are independent of specific cases, these systems are more general. A conflict rises because of the need for efficient domain specific expertise [van de Ree, 1994]. General knowledge is inefficiently applicable to most domains; specific domain expertise cannot be used for other domains. A careful balance is necessary.

3.3 Machine learning techniques

Besides the shallowness, a limitation of the first generation expert systems is that they do not learn. The knowledge is only analysed, not synthesised. In recent years much research has been performed into machine learning (ML) techniques, in which it has been attempted to make programs improve their performance. This is
interesting both for instructional scientists and for AI scientists. Models for human learning strategies are composed, which could be used for teaching purposes. For AI researchers, it is interesting to make machines do what people had almost stopped believing to be possible: make machines learn. This section gives an overview of promising ideas in this field. The overview is meant for people not familiar with machine learning concepts; it gives a first introduction without going into much detail.

3.3.1 What is learning?

Very much has been written about learning, and many definitions can be found in literature. Most of them incorporate the criterion of improving performance. The principles, however, are still not well understood. An elegant definition, by the machine learning scientist Michalski, is

\[ \text{Learning is constructing or modifying representations of what is being experienced [Michalski, 1986].} \]

Rather than improving performance, the central aspect is the process of constructing a representation of some reality. Performance improvement is considered a consequence of building the representation. In more recent work, Michalski states

\[ \text{Learning is a goal-guided process of modifying the learner's knowledge by exploring the learner's experience [Michalski, 1993].} \]

Much knowledge is generated by the generalisation of observations. This is inherently conjectural; it cannot be proved correct, but it may be disproved [Michalski, 1986].

Expert systems only use deductional inferences. With fuzzy systems, also the concept of abstraction is used, i.e. discarding details not necessary for reasoning. People also use other types of inference, such as induction and analogy.

**Inference types**

The fundamental equation for inference is

\[ P \cup BK \vdash C \]  

(3.1)
In this equation, P is the premise, BK is background knowledge, and C is the conclusion. Deduction is the process of finding C when P is known. Induction is the process of finding P when C is known. Induction preserves falsity, deduction preserves truth. For example: take the statement "a crow is a black bird". Deduction derives that an animal is a black bird when we know it is a crow. Induction hypothesises that an animal is a crow when we know it is a black bird.

Inference by analogy is a very common kind of conclusion. For people, reasoning using analogs is a natural process. For experts, the use of analogs in reasoning results in reliable solutions. For instance, we know that the solar system consists of a large object in the middle with smaller objects circling the large object. While assuming that an atom consist of a large nucleus and small electrons, a reasoning by analogy is that the electrons are circling the nucleus.

Analogy is somewhere between deduction and induction. It does not preserve truth, because analogies can be wrong, but it also does not preserve falsity.

Inference patterns

A concept F is more general than a concept G if it covers a larger set of objects. An example of generalisation is to derive a rule from a set of examples:

\[ P(a_1), P(a_2), \ldots, P(a_n) \Rightarrow \forall x P(x) \]  

(3.2)

Specialisation is usually deductive; from general knowledge, a specific case is derived:

\[ \forall x P(x) \Rightarrow P(a) \]  

(3.3)

Abstraction is applied when a description is too detailed for future inferences. It is used, for example, in fuzzy sets: a variable is classified into certain categories, such as certain lengths being short, normal or tall. The transformation "Cornelis has a length of 1.96 meters" to "Cornelis is tall" is called an abstraction. With concretion, more detail is added if the information is not sufficient. In the same example, if we want to know whether Richard or Cornelis is taller, knowing they are both tall, we need concretions of both lengths.

Similisation is used to obtain more knowledge on the basis of similarity with other objects about which more knowledge is available. Dissimilisation derives new knowledge on the basis of lack of similarity.
3.3 Machine learning techniques

There are far more patterns of knowledge change; Michalski handles these knowledge transmutations, as he calls it, in his inferential theory of learning [Michalski, 1993].

Bias

If background knowledge is taken into account, the number of possible hypotheses can be enormous. There has to be a selection criterion to limit the choices. Usually we are only interested in justifiable, simple or plausible hypotheses. The selection criterion is also called bias. A chosen implementation of a learning program can also cause a selection criterion, known as description language bias.

3.3.2 Learning from examples

A well studied subdiscipline of ML is learning from examples: teach computer a concept by showing positive and negative examples. The teacher selects examples that are representative and provide a good description of the concept. For example, when teaching the concept of bird, an ostrich or a penguin is not a good example to start with, because they do not fly. Negative examples are used to draw attention to features that differentiate between instances and non-instances of the concept; they are usually “near misses”.

Several algorithms have been developed for learning from examples. The algorithms mentioned in this section are from Kubat [Kubat, 1992]. One of them is the AQ algorithm, known since 1969. In it simplest form, it works as follows:

1. Classify all examples into a set of positive examples and a set of negative examples.
2. Choose a random positive example.
3. Find the most general description(s), making sure that the generalised description does not incorporate a negative example.
4. Find the most desirable of these descriptions.
5. Stop if the description covers all positive examples. Else choose another random example and resume at step 3.

Another well-known algorithm is Top-Down Induction of Decision Trees (TDIDT), better known as ID3. The input is equal to that of the AQ algorithm; it produces a decision tree usable for the classification of new examples. The algorithm is:
• If all examples in the training set are positive examples of T, then the decision tree for T consists of a leaf labelled with T.

• Otherwise, choose a test that partitions the training set into two or more subsets. This test becomes the root of the decision tree. For each outcome, build a subsidiary decision tree by invoking the same procedure recursively on the subset.

An important factor is the choice of a proper test. Usually this choice is based on information theory. If the examples are described by a number of attributes, each attribute partitions the example set into a number of classes k (this number is equal to the number of values occurring in the attribute). The attribute offering the highest information gain is taken. The information gain can be defined by means of the entropy H for an attribute, shown in Eq. 3.4. In this equation, \( p_i \) is the probability of class i, \( p_{i,pos} \) is the probability of a positive example in class i, and \( p_{i,neg} \) is the probability of a negative example in class i.

\[
H = -\sum_{i=1}^{k} p_i (p_{i,pos} \cdot 2 \log p_{i,pos} + p_{i,neg} \cdot 2 \log p_{i,neg}) \tag{3.4}
\]

H is the weighted sum of the entropies for all classes; the entropy for a class is calculated by Shannon's theorem. The attribute with the smallest value for H is chosen.

There are many more sophisticated methods capable dealing with noisy data. One of the interests is pruning trees to improve classification accuracy [Bratko, 1994]. Learning by examples assumes there is a "teacher" saying which examples are good and which are not. Giving training is not a question of pure classification on the basis of attributes. The "attributes" are not known in advance and values can change in time; questions can reveal weaknesses which were assumed to be available, for example.

### 3.3.3 Learning by observation

In learning by observation, the goal is abstracting knowledge from data. Before AI, this field was mainly covered by statisticians, with cluster, factor and correlation analyses. These methods are often inadequate when using non-numerical data. Further, there is no way to express knowledge about the domain, and the results are not easy to interpret. Clustering data into categories still does not say anything about the concepts underlying the categories. These and more arguments were reasons for
much research on this field in AI. A currently often used term for this research is data mining, or automatic inductive reasoning conditioned by data. Further, the term conceptual clustering is often used.

More specifically, the aim is to find clusterings which define new disjoint concepts, given a set of non-classified examples (described by attribute-value pairs), an evaluation metric, and background knowledge. Learning by observation can be seen as unsupervised learning, whereas learning from examples is supervised learning.

One way to classify conceptual clustering systems is by their method of dealing with data: incremental or non-incremental. A non-incremental system is Michalski’s CLUSTER/2. The algorithm is shown in figure 3.2 [Bisson, 1994]:

A star is defined as the collection of all most general descriptions of a seed (a starting element). The optimization process is aided by a user-defined evaluation function in which, for example, criteria as the number of variables that discriminate among all clusters, the simplicity of the resulting clustering, or inter-cluster difference can be applied.

COBWEB uses an incremental method. The principle is that an elementary motivation for categorisation is to be able to predict category features. COBWEB builds a hierarchy of probabilistically described concepts. There are two important opera-
tors; one splitting an existing concept into two, at the same level, and one merging two concepts into one. Merging means pushing down the concepts one level and creating a new concept covering the merged concepts.

The decision as to when to use an operator is based on an evaluation function, measuring the increase in the expected number of attribute values that can be guessed correctly using the categories, over the expected number of correct guesses without this knowledge [Kubat, 1992]. Every time a new example is read, the concept hierarchy is adapted by adding new nodes, splitting a node or merging two nodes. The hierarchy is traversed until a node results with only one example, or until an evaluation threshold cannot be met.

The conceptual clustering theory is meant for classification as well, making it less likely to be suitable for a training system. However, a probable application is the analysis of session data, for classification of students based on their results (and compared to others) and for checking the difficulty estimations of items after sufficient data is available. This is more difficult with the multidimensional approach advocated in this thesis than with the unidimensional IRT method. This could well justify a less purely statistical approach. This is an area which still needs to be researched.

### 3.3.4 Case-based reasoning

The idea behind case-based reasoning (CBR) is to learn and reuse concrete experiences instead of (primarily) general knowledge. Schank and Slade formulated this as follows:

> In predicting the future, we look to the past. When faced with a new problem to solve, we are often reminded of previous similar episodes, the solutions to which may be adapted to the current problem [Schank & Slade, 1991].

Motivation for research in case-based reasoning (CBR) comes from cognitive science on one hand, as a theory of understanding, problem-solving and learning in human beings, and from knowledge-based systems research on the other hand, because of the deficiency of purely generalisation based methods for intelligent computer programs [Aamodt, 1994]. Usually CBR is used for classification tasks. Domains which are especially suitable for this approach are those which require diagnosis, for example, medical diagnosis, oil well drilling, and help desk systems. Some experiments have also been done in criminal justice. Bain experimented with a system called JUDGE, designed to find proper sentences for crimes, taking into account possible jurisprudence [Bain, 1986].
A CBR cycle has four phases:

**Retrieve.** When given a new problem, retrieve a similar case from memory. Similarity is stored by indexes.

**Reuse.** Make use of the solution to the retrieved case. Substitutions and transformations or a derivational replay can be applied.

**Revise.** Apply the solution to the problem, evaluate the results, and repair possible failure. Often this reparation is done by asking the user, sometimes the system has its own possibilities.

**Retain.** Extract relevant information such as how to describe the problem (indexes), the solution and the solution method. Then learn the indexes, store the case, and possibly update general domain knowledge (e.g., structure of categories).

There has lately been much interest in CBR, as can be seen from the many commercial tools appearing on the market [Aamodt, 1994].

Case-based reasoning does not seem really suitable for a general training system. There is no intuitive way of specifying the cases. One way would be making the student into a case, with, as attributes, among others, the estimated proficiency levels. The case solution is a suitable subject and level of difficulty to ask a question about. However, this is not a very practical method as it would require a large storage facility for cases, and an initial database, or a method to calculate the result if no cases are available. It is probably more useful to use a CBR method to diagnose error causes with students, as teachers have noted that often the same problems appear with every group of students. This also would have to be examined in more detail.

The method, as a model of thinking, can be used for educational purposes. For example, a story telling system was built [Bell & Feifer, 1992], based on task-centred instruction.

### 3.3.5 Reinforcement learning

When we look at human beings, the vast majority of skills are learned by trial and error, rather than by being "programmed" how to do them. Research into reinforcement learning is focused on autonomous systems that learn the details of their behaviour from high level feedback about the world.

A description of reinforcement learning is given by Kaelbling, who was also the source for most of the information in this section:
Reinforcement learning is a popular model of the learning of problems encountered by an agent that learns behavior through trial and error interactions with a dynamic environment [Kaelbling, 1994].

The model is shown in figure 3.3 [Kaelbling, 1994]. The agent is visualised as a mobile robot. The world situation is filtered by an input filter I and a reinforcement filter R. The agent must map each situation (or information state) into an action to be performed in the world. The reinforcement tells the agent how well it is doing. The goal is to maximise the total reinforcement it gains over its lifetime.

The difference between this type of learning and concept learning is that with reinforcement learning, a behaviour is expected from the beginning, and there is no presentation of examples. Further, the environment is usually noisy. Reinforcement learning is also called "learning with a critic" in contrast with the supervised learning method of "learning with a teacher".

A successful example is learning backgammon. An artificial neural net (see section 3.3.6) is used to map inputs to actions. At the moment, top players have a hard time beating TD-Gammon, as it is called; after 1.5 million training games it lost only one point in 40 games against top players. Another example cited by Kaelbling is a mobile robot for mail delivery, with as goals to navigate down corridors via sonar, and to build a map of the building.

The amount of training needed shows a drawback: if taken alone, reinforcement learning methods are likely to be impractical. A vast amount of information is needed to learn a behaviour when there is no basis of previous knowledge. A more practical system would incorporate a priori knowledge with on-line learning.
Is it possible to see "giving training" as a reinforcement learning problem? Apart from implementational issues, such as how to define a meaningful reinforcement, the drawback arises that during the first (possibly quite long) period, behaviour will be virtually random. This is not very desirable.

3.3.6 Biologically inspired methods

Artificial Neural Nets

Artificial neural nets (ANN) have regained popularity lately. The basic component of a neural net is shown in figure 3.4. The output $y$ is often calculated by a weighted sum of the inputs $u$, followed by a non-linear function. If the function is a binary threshold, the basic component is called a perceptron. ANNs are formed by networks of these neurons, usually consisting of three layers: the input layer, receiving and passing on input, the hidden layer, transforming the inputs and passing them on, and the output layer, receiving and transforming the signals from the hidden layer and passing them into an output vector. See figure 3.5.

The most common method of training an ANN is supervised learning, in which the weights are adjusted by the learning process, using examples of input-output vector pairs. An often used method is error backpropagation, in which the difference between the obtained output and the desired output is used to update the weights. By doing this first for the output layer, the desired behaviour of the hidden layer can be reconstructed, which is used to update the weights from the input layer to the hidden layer.

A big advantage of ANNs is the capability of learning an arbitrary non-linear function. However, it cannot easily be said what exactly the ANN has learned; its knowledge is hidden in the weights and is not known explicitly. Further, it may take a lot of teaching before the ANN performs well.
There are many successful applications of neural networks. One of the most famous is NETtalk, developed by Sejnowski and Rosenberg in 1987. NETtalk learns to pronounce English text, it has achieved about 90% correct phoneme pronunciation. The opposite has also been done: a phonetic typewriter was developed in which neural nets were used for the classification of phonemes. Other applications are pattern recognition, airline marketing, ECG noise filtering, and financial forecasting.

Neural nets are usually applied if an unknown or difficult to analyse non-linear function is needed, for which inputs and outputs are available. The net is trained to mimic the function. Although some nets are generalising, there is no guarantee that unseen inputs will give reasonable outputs. This is a great drawback if it is applied in a training environment. Further, with training there is a model, only the parameters are not known. So neural nets do not form a logical choice either.

**Genetic algorithms**

Genetic algorithms (GA) can be seen as a software implementation of the evolutionary theory. According to this theory, evolution in nature is based upon a trial-and-error mechanism, known as “survival of the fittest”: what has been successful has a higher probability of surviving. The word “fittest” denotes a criterion which has to be maximised. Genetic algorithms can be used, for example, to find the maxima of functions which are too complicated to use an analytical solution. The basic idea is shown in figure 3.6.

The algorithm works as follows [Kubat, 1992]:
In the *initialisation* phase, a set of randomly selected numbers, in the form of bit strings, is chosen.

In the *reproduction* phase a new set is generated. From the old set, strings will reappear in the new set with a probability proportional to their functional value. Strings with high values can reappear several times; strings with low values are likely to disappear.

The *mutation* phase occasionally changes a random bit of a random string.

The *recombination* phase consists of recombining the strings. A simple method of doing this is single-point crossover, in which the last k bits of two strings are exchanged. This set is the basis for a new iteration, starting with reproduction.

The reproduction phase selects which candidates are fit enough to survive. The recombination phase is the driving force towards the maximum, trying out neighbours of good candidates. The mutation phase prevents the algorithm from getting stuck in a local maximum by giving a chance to candidates not in the neighbourhood of a current candidate.

The interpretation of the bit strings can vary. In the maximum finding example a bit string represents a number. A bit can also represent the occurrence of a certain attribute, which makes a bit string represent an entity or a concept. More generally speaking, it is not necessary to represent the world in bit strings. For example, another representation is in the form of trees; mutation is then implemented as a change in node value, and recombination as exchange of sub-trees of two candidates.

The function to maximise specifies how well the entity is doing. Natural evolution seems to continue endlessly. For technical algorithms, a termination criterion can be applied, for instance, “stop if the maximum has not changed in the last x generations”.

Figure 3.6. *The principle of genetic algorithms*
Artificial intelligence and computer-based training

The weaknesses of genetic algorithms are that there is no theoretical framework for the evolutionary process, there are no guarantees that a solution will be found, and progress is intermittent, making it hard to predict the type of solution and the time it will take to find it. However, the strengths are that GAs are less vulnerable to local optima, and that the search is opportunistic, often causing solutions to be "pleasantly surprising" [Haasdijk et al., 1994].

Problems for which generic algorithms can be useful are optimization problems when not much knowledge is available. For use in educational systems, the same limitations as apply to reinforcement learning hold for genetic algorithms: it takes a long time before results are usable. This problem could be solved by using "virtual" or "simulated" students until the results are usable, but then a lot depends on the similarity between real students and virtual students. Even then, problems remain, such as how to define the genetic code.

3.3.7 Discussion

The aim of a training system is assessing the student's knowledge in order to improve the student's performance. The student's knowledge is assessed by interpreting answers to questions, the student's performance is improved by asking questions that are challenging, keeping the student interested. Two key elements are the making of a model of the student's abilities from the items and answers (and possibly prior knowledge), and selecting a new item based on the model. Both elements are not classification tasks in the sense that objects with known attributes have to be placed into categories of similar objects. Therefore, machine learning methods in that field are not very helpful. Further, other methods like genetic algorithms, artificial neural nets and reinforcement learning methods start to exhibit reasonable behaviour after a considerable learning time. Often the behaviour is almost random in the learning period. However, using a non-ML method, the system performance will not improve.

It is very likely that machine learning techniques will lead to more sophisticated educational systems in the future, maybe even in the near future. In ITS research especially, there are many opportunities. However, at the moment, for a practically usable training environment, "normal" adaptive methods seem more appropriate.

3.4 Blackboard architectures

A blackboard architecture offers an environment for distributed problem solving. The components of a blackboard architecture are:
3.4 Blackboard architectures

- Knowledge sources
- Blackboard data structure
- Control mechanism.

The architecture is best described by a metaphor. Imagine a number of independent experts cooperating to solve a problem. The experts have expertise in different fields relevant to the problem. They use a blackboard to write down the goals and the intermediate results. If an expert is triggered by information on the blackboard, he can act and give some new information on the blackboard.

A chairman can be assigned the job of keeping structure in the solving process (see figure 3.7).

![Figure 3.7. Blackboard architecture metaphor](image)

In such a situation, an expert informs the chairman if the information on the blackboard gives possibilities for the extraction of new results. The chairman allows the expert that has the best chance of coming closer to the desired goal to use his expertise. Of course, this is a simplification of the process. A more elaborate handling of blackboard architectures and systems can be found in [Jagannathan et al., 1989].

Advantages of a blackboard architecture are

**Modularity.** The problem to be solved is split into independent sub-tasks.

**Concurrency.** Because the sub-tasks are independent, they can work concurrently on a solution.

**Efficiency.** Problem data is stored only on the blackboard, there are no multiple presences of the same data. Further, only tasks most probably leading to the goal are executed.
Artificial intelligence and computer-based training

Care should be taken to prevent deadlocks in parallel situations. Further, control of the blackboard control must be arranged well. If tasks are able to change data “at random”, a truth maintenance system will be needed. Otherwise, the freedom of modifying data must be restricted.

Blackboard architectures are especially appropriate for

- Integration of disparate knowledge
- Natural domain hierarchy
- Continuous data problems
- Applications with sparse knowledge/data
- Real time problem solving.

For the architecture of a general training system, a blackboard system is appealing. The goal of the system indicates that there are several independent agents [Essenius, 1994]. A blackboard system allows separate expertise units to cooperate. Further, it stimulates a modular approach. If structured correctly, it must be possible to replace a unit by another one easily, implementing another strategy or another method.

3.5 Conclusion

Concerning the usability of AI techniques, machine learning techniques are not the most likely candidates for use in the system. ML is especially useful for classification tasks, data mining and knowledge acquisition. These are not the key issues in a training system, although classification tasks can be distinguished. However, the problem is not classification, but the acquisition of enough information from the student.

A model-based approach seems more appropriate. A flexible basis for the system is a blackboard architecture. This allows a modular design of all functions needed although much information is needed in more than one function. An object-oriented approach could provide an elegant realisation, by using messages instead of directly changing data.
A general computer-based training shell

4.1 Purpose

The aim is the development of a platform-independent training system, offering students the possibility to practice their ability on a course. The practice consists of problem-solving activities, intended to lead to abilities comparable to those required in the course examination. After a training session, students must be given an impression of their ability. They can use this impression in order to identify possible weaknesses needing extra attention, for example.

To improve effective time, the training should be at an appropriate level for the students. In other words, adaptation to the student's level is desired. In a later stage, the adaptive program control may be replaced by a student advisory control. In both cases the adaptive mechanism plays an important role. Therefore, much attention will be given to models needed for adaptation.

This chapter has the following structure: first basic assumptions are discussed, followed by the chosen design approach. Then the analysis, design and implementation phases are described.

4.2 Basic assumptions

Most courses in engineering education can be divided into several subjects with a certain taxonomy, the most obvious relations being prerequisites. This taxonomy defines the curricular model for the course. It is assumed that this model is available for the courses with which the system is used.
A general computer-based training shell

Usually, when a teacher examines a student orally, he/she first samples the proficiency roughly by asking a few questions; after that the results will be taken into account when asking new questions. Answers to items in a current subject can show possible problems in prerequisite subjects, so a question in that direction can give better insight into the student's proficiency on the subject. Of course, an examination is different from a training session, but the observation that answers to items can give information about the students' level on other subjects is still valid.

This observation shows that items, intended to test proficiency on a specific subject, can reveal information about other subjects. In other words, items themselves can be multidimensional.

The system must be domain independent, i.e., independent of the course it is used with. In other words, there is no explicit domain knowledge available.

With the intended general applicability in mind, it is reasonable to assume that the system will be used by more than one teacher. Every teacher has his/her own ideas about the best way to give training; therefore if the system has a rigid strategy, it is likely that it will not be used on a wide scale. The strategy should therefore be flexible. For example, one teacher could use the incremental strategy, in which the students start at the bottom with the basics, and work their way up to higher subjects when mastery in the lower part is shown. Another strategy is to sweep through all subjects a number of times; this approach is advocated by people who think repetition after different subjects is important. The first sweep can bring the students to a level in which they know enough for the next subject, the second sweep can take them to mastery. A third teacher could give the students more freedom by allowing them to influence the decisions. Also at the subject level, flexibility is required for proper usability. If a difficulty estimation of items is possible, a strategy such as IRT could be applied, whereas if no difficulty ordering is available, SPRT or the Beta method is more appropriate.

By offering this flexibility, maybe it will be possible to bridge the gap between theory (interesting research) and practice (deliverable instructional systems), by giving opportunity for here-and-now items in simple multiple choice or numerical format, but keeping open the possibility to incorporate more sophisticated items with an ITS character, for example to replace items on a specific subject.

This last statement reveals another assumption for the system: it must be flexible with respect to usable items. Using the item implementation strategy from Items-D, i.e., let the teacher decide for himself what environment to use as long as the resulting item is executable and the item can communicate with the system, offers this flexibility. Another advantage of adopting the strategy is that the item banks made for Items-D will be directly available.
In conclusion, the basic assumptions are

- Courses can be modelled by a taxonomy of subjects, each subject can be assumed unidimensional.
- Item selection strategies must be changeable, both on the course level and on the subject level.
- Available items, made for Items-D, must be applicable.

### 4.3 Design approach

With the basic assumptions of flexibility in mind, an object-oriented design method [Aksit et al., 1992] seems a suitable way to structure the process. By identifying the models as objects, it becomes easy to exchange an object with another instance performing the same task in a different way.

When building a system, three phases can be specified. The first is the *analysis phase*, in which a correct identification of user’s needs is given in an understandable way. It results in an analysis model, identifying the problem domain and system responsibilities. The second is the *design phase*, in which the analysis model is revised and extended by specifying how to realise the requirements. These phases are described in the following sections. The third phase is the *implementation phase*, in which the design model is detailed by means of language constructs. The description of this phase is limited to a justification of the chosen programming language.

### 4.4 The analysis phase

Initially, the goal specification was not entirely complete. After several iterations with teacher feedback an analysis model was formed. This section describes considerations leading to that model.

The system must offer the students a training session in a course. The goal of the program is to lead the student through the course, by offering items, i.e. problems to solve.

Key issues in this description are the concepts of course, student and item. We start the analysis by modelling these candidate objects, investigate whether they are suitable and if other objects are needed. Finally, relations between the chosen objects are identified.
4.4.1 The models

Course model

The course consists of several subjects which are related to each other via prerequisite links. This taxonomy must be alterable without having to change the program. In other words, the taxonomy is entered into the system at run time, most probably at initialisation. The actual implementation of the taxonomy must be hidden, to allow different versions and different selection mechanisms. For example, the links may be only and-links, meaning that all prerequisites will have to be on a certain status before the subject can be chosen. Another possibility is that also or-links are allowed; this means that one of the prerequisites is enough to make the subject eligible.

The most important action to be performed on the course model is the high level selection of a suitable subject. This selection uses the status of low level student models — e.g., a measure of confidence regarding mastery in a few discrete values — as described in the next subsection.

Student model

The student model is a reflection of the course model: it contains a specification of the student's performance on each subject in the course model. These low level student models are called subject models. The low level student model is used to determine a proper criterion for item selection.

For example, with SPRT, the low level model would be the number of items answered and the number of items answered correctly, with IRT it would be the estimated proficiency distribution. If wanted, final student models of previous sessions, or student preferences, can be used as a starting value for a new session. If these are not available, teacher-specified defaults must be used.

Another action to be performed on the student model is the updating process after an item has been answered and a response is available. Low level updating concerns maintaining the performance and, with that, the status. In this context, the low level model maintenance could be seen as a monitoring task, and the high level model maintenance as a diagnostic task.
Item model

The item model can also be divided into two sections: the high level model, consisting of subjects possibly influenced by the item and criteria for influencing, and the low level model which is a reflection of the chosen low level unidimensional method. Again taking the IRT method, the low level model would consist of the item characteristic curve specification for the item, or, more specifically, the difficulty, discriminating value and guessing correction.

Another item parameter is the series identification. In many courses there are items of a different level of difficulty, and even items on different subjects, that are still of the same series of items. When a student enters one of those series, he should remain in them if possible. An example in the field of Electronics is a series starting with a simple electrical circuit. Several increasingly difficult questions could be asked, the circuit could be extended, etc. In theory, it does not matter very much which circuit the student gets, but in practice the student needs some time to comprehend the circuit, and when the following items deal with the same circuit, this time is not needed for each item.

Responses to items can have different formats. For model updating, the literal answer is not necessary. Some student model updating methods do not need more information than the correctness of the answer. For some methods, the time used is taken into account. In other items, the chosen alternative in a multiple choice item—or the answer category with other anticipating items—is used to identify subjects that need to be updated, besides the current subject, and in which way they need to be updated. Consequently, the item response can be multidimensional, influencing more than one subject in the course.

4.4.2 Control

Actions will be coordinated according to a control scheme. The least form of student control is that a student can enter the desire to stop every time an item has been finished; to allow students to make more founded decisions, the impression about their abilities must be shown as well after ending an item. The main target for the system is adaptive program control, although there should also be room for student advisory control, in which the system suggests an action that the student can accept or not, or in which the system suggests several actions of which one can be chosen.

Different control strategies will result in different use of the models; often the effects of decisions cannot be forecasted exactly. Sometimes backpropagation of results takes place, causing other actions than initially planned to be necessary. For
example, if an item selection fails in an adaptively controlled situation, it is apparently not possible to find a suitable item. This means that the chosen subject is no longer available, and another subject must be chosen. However, if, in a student advisory controlled situation, the student decides to disagree with the advice and asks for another item (for example, more difficult), the subject may remain valid. In that case, a new item selection can be done directly. The person responsible for implementation of the control strategy should be bothered as little as possible with these problems; they should be solved below the surface.

4.4.3 The item interface protocol

Items are made externally and incorporate all subject domain knowledge, usually implicitly. This means that also possible feedback and solutions are handled inside the item. An item can be seen as a template in which some parameters have to be filled in to present a meaningful problem. One can think about parameters in the item itself, to be able to give different variations of a particular item every time it gets chosen, and meta-level information, for example how many points the student can get at maximum, how many attempts the student may use and the score-subtraction for asking help or needing a second attempt at answering. Also the right answer to a problem is not always known at the moment the item is written. Therefore the session manager must be able to transfer this information to the item at the moment it is started. When the student has finished the item, information about the performance will have to be returned to the session manager. To be able to achieve this communication, a protocol was specified and implemented in several environments. In the following three subsections, the requirements for the interface and the higher and lower level protocols are described.

Requirements for the interface

The aim of the interface is a flexible communication between session manager and item. What exactly do we want from the interface? First of all, it must be applicable to a wide variety of implementation environments and portable to other platforms. Second, the information flow from the session manager to the item must contain the needed parameters, correct answers, maximum achievable score and factors which could influence the score. If the system is to be used in a training environment, it will have to be able to anticipate common errors in order to offer the student assistance, for example in the form of a hint.

When using a non-adaptive session manager, for example with sessions in which the items can be chosen from a menu, it is allowed to interrupt the answering of an item
and resume afterwards. To allow this, the item must pass the current status of the item, the status of all questions in the item (number of attempts used until now, whether help was asked, etc.), and possibly some other intermediate information. This will make it possible that the next time the item is chosen, it restarts with the right status and at the right position.

Items can use other software for complicated calculations or simulations. A very common way to do this is defining a command script which will be executed by the external program after start-up. Parameters for this external simulation could depend on parameters which are offered to the item, making it impossible to completely specify the command file beforehand. So either the item must be able to complete the script files after the desired parameters are known, or the interface must provide the script.

The protocol principle

Taking into account the desire for multi-platform and multi-environment availability, a safe and flexible way is the use of ASCII files. On every platform, it is possible to generate and read ASCII files (or at least to convert them to understandable formats). The only limitation this poses is that the development environment must be able to read and write ASCII files. For general programming languages, this is no problem, and also a lot of authoring languages support it. Most systems not supporting it directly offer a possibility to link home-made code so it can be realised after all. A disadvantage of using files is that the (relatively slow) file system is used, but speed is not a very critical factor as the files are only accessed just after user input.

The communication protocol was adopted from the item banking system Items-D [Jansen et al., 1991]. This was chosen to make items created for Items-D compatible with the new system. This protocol uses two data files, the item information file and the item result file. The first is created by the management system just prior to starting a new item. The item will read this file at initialisation. Each time the student gives an answer, the item result file is created or modified by the item. At termination of the item, the management system reads and processes this result file.

One of the features of Items-D is that it allows parameter variations in the items, to accomplish that two students receiving the same item get different numerical variants. This variation of parameters — internal to the items— must be supported by storing specifications with the items in the item bank. The interface specification of Items-D requires the use of information files with a specific name. This means that items must be executed in a unique directory, to prevent files from being overwritten in a multi-tasking environment.
The interface files

The item information file consists of three sections. The first is called the parameter section, specifying the variable parameters in the item which are needed to obtain the correct answer. Parameters can be of various types including integer, real, Boolean, and text. The second is the answer section, specifying answers to all the (sub)questions in the item, and their maximal scores. Also incorrect or partially incorrect answers can be anticipated. The third is the miscellaneous section specifying other information about the item. It can contain, for example, subtraction percentages for wrong answers or for requesting assistance. Further, it can contain text sections if the item is a template, and file specifications if the item needs text files for proper execution. These files can be command files for external simulation or calculation software, or for the display of graphs.

The item result file contains the statuses, answers and scores of all subquestions in the item, together with a summary and a miscellaneous section showing which of the score-influencing factors was used. Also some temporary information can be stored.

The interface specification is elaborated in detail in Appendix A, sections A.1 and A.2.

4.4.4 Platform-dependent objects

The aim of platform independence can be relaxed into a requirement that platform dependencies must be grouped together. The user interface is an example of a possibly platform-dependent object. Problems can occur there if nonstandard actions (for example, moving the cursor to a specified position, or changing the screen or character colour) are needed. To obtain a clear user interface, this is inevitable. By keeping these actions within the user interface and allowing objects to perform only standard actions, the changes needed to support another platform are minimal. Information feedback to the student consists of the current student model, an item history (in the most simple case, the last item finished), and the intended next item. The user can make choices from a number of options (for example, to indicate whether or not to continue) and enter values or strings.

The sessions must be logged, for example to allow statistical analyses. It is thinkable that with low level models such as IRT, automatic item model updates can be scheduled, when enough data is available to fine-tune the item bank. First of all, the final student model is stored when a session is ended. Further, every time an item has been finished, the item characteristics as well as the changes in the student
model are logged to a file. Since the system must be able to run in a multi-user environment, these log files must have unique names or unique directories. For later gathering, unique file names in one directory are preferred. The session logger shields the platform-dependent functions and file details from the objects to be logged.

Also accessing other directories is platform-dependent; the file management functions should be controlled by a single object, the file manager, again in order for the other objects to need only standard I/O functions.

Since the program can be used for various courses, a mechanism must be designed to be able to indicate the desired course. This will most likely be platform-dependent as well.

Care should be taken that the user interface and the file manager do not need too much information. For modularity reasons, it is not desirable that they need access to the student model, because that would imply that they need alteration if another model is used. In other words, all objects needing user I/O or logging output must know themselves how they can read or write their information, if given a stream to read from or write to.

### 4.4.5 The analysis model

The most important objects are shown in figure 4.1. The item interface is not mentioned here because it can be encapsulated in the item manager, which is the object responsible for all actions involving items. Also the session logger does not have to be a separate object; it can be included in the file manager.

![Object specification diagram](image-url)

*Figure 4.1. Object specification*
A general computer-based training shell

Arrows denote use-relations (the object pointed to is used); the line with an open arrow in the middle is a part-of relation. This particular one shows that a course model can have one or more subject models, and a subject model has exactly one course model.

The use-relations can be translated into methods related to the object used:

Course model:
- want subject
- update student model
- log progress
- show model

Item manager:
- select item
- execute item
- want item results
- log item
- show selected item
- show finished item

File manager:
- want list of courses
- use course
- want to read course structure
- want to read student model
- want to write student model
- want to read item manager data
- want to write item manager data
- want to read item data
- want to write to log file
- done writing/reading

User interface:
- want to show item choice
- want to show result of item execution
- want to show course model
- show message
- process menu
- want to ask question

With this model, the problem domain and the system’s responsibilities are specified. In the design phase, the model is elaborated in more detail.
4.5 The design phase

The analysis phase showed a number of important objects; these are expanded on in this section. Human interaction, task management and data management will receive more attention.

4.5.1 The course model

The course model is built up from one or more subjects. First the subject model is described in more detail, then the actions to be performed at a higher level.

The subject model

All the unidimensional models described in Chapter 2 have a structure as shown in figure 4.2. The main difference between a stand-alone unidimensional system and a multidimensional system is that the higher level model can intervene if another subject is given more priority. For this, the status is introduced; it allows giving information about the student’s progress, without going into detail regarding the implementation of the subject model. The stop criterion is translated into a certain value of the status. Consequently, part of the model update process is updating the status. This updating process resembles the calculation of the former stop criterion, albeit that there are more possible values than the usual pass/fail or enough/not enough information. The update process will use the given low level item model (if any) and item score.

The item selection strategy differs greatly per model. With some, an item is selected randomly, but others select items based on a certain criterion, for instance items
with a difficulty as close as possible to the current estimate. IRT even has a strategy
in which all items have to be tested to see which is best. This last method is not very
efficient, and it is possible to make a smarter implementation by defining an info
table, giving a number of appropriate items, in decreasing order of desirability, for
each value of proficiency. This info table has to be modified only if new items are
added to the system, or if item parameters are changed. Creation of the table can be
left to the program. With this implementation, with IRT it becomes possible as well
to give a single criterion on which to select an item.

A problem with some of the described unidimensional models is that it can take a
long time before a decision has been reached. This is not always desirable, often it is
better to limit the number of items to be asked. Whether or not this maximum has
been reached can be included in the status.

The subject model must make the information available for the outside world. It can
be saved for a next session, or shown to the student. If it can be saved, it must be
possible to reload it as well. Summarising, the object becomes:

**subject model**

attributes
- maximal number of items
- proficiency estimation
- status

services
- update model (in: item, response)
  - recalculate proficiency estimation
  - recalculate status
- want status (out: status)
- want item selection criterion (out: criterion)
  - calculate criterion, if needed
- write model (in: out-stream)
  - save situation (attributes, model)
- read model (in: in-stream)
  - reload last situation
- show model to user (in: out-stream)
- log model (in: out-stream)

**The subject**

The course model consists of a taxonomy of nodes called *subjects*. The subject
model cannot be used directly because the node contains information that must be
hidden for the subject model: the prerequisite information. Therefore the subject
model and the prerequisites have been made parts of subject, see figure 4.3.
The subjects further have a name for easy recognition by students and an identification number by which they are known in the item bank. These are the attributes.

**subject**
- attributes
  - subject identification
  - subject name
- services
  - show (in: out-stream)
    - *show the current subject data to the user*
  - log (in: out-stream)
    - *log the current subject data*
  - available with state (in: astate; out: available?)
    - *return true if the subject has state astate and is available*

When created, the subjects receive their name, identification and prerequisites, which remain constant throughout the session.

**The course model**

Now that the node specification of the taxonomy is available, the structure itself can be specified. Besides the prerequisite information, there is also a desired curricular order. This order is specified by the identification numbers of the subjects: the lower numbers are earlier in the course. An important aspect of the course structure is that it can be changed without having to recompile the system. During a session, the structure will remain the same. This implies that the structure can be read from a stream during system initialisation. After that, possibly present student models and update parameters are read or defaults are fetched.
Course model

services

- initialise
  - retrieve course structure and initialise student model
- want criteria (out: subject, subject level criterion)
  - return appropriate subject and criterion to ask about
- select subject (out: success?)
  - select a subject and return if the selection succeeded
- update (in: item response structure)
  - update the student model according to the structure
- set search failure on current
  - indicate that no item is available for the current criteria
- log progress (in: out-stream)
  - write the process progress to the stream
- show model (in: out-stream)
  - write a heading
    - for all subjects:
      - write subject model to user

4.5.2 The item manager

The item manager (see figure 4.4) is responsible for all actions involving items. It contains a list of possibly influencing subjects (including the main subject), corresponding item models for those subjects, and influence criteria. The item models depend on the chosen subject model. Further the item manager contains an answer model, current item series identification, and attributes needed to be able to execute the item: the run environment, the item file name and the run parameters.

![Diagram of the item manager](image)

Figure 4.4. The item manager
4.5 The design phase

**Item manager**

attributes
identification
run parameters
run environment
file name
current item series

services:
initialise
read item manager initialisation data
select item (in: subject, select criterion)
use subject & criterion to restrict number of items
read information about applicable items from item bank
if unused items of current item series exists
- select one
else if unused items exist
- select one
- update current item series
else if items exist which have not been asked too often
- select one
return true if an item was selected
execute item
get parameter & answer data from data base
choose parameters
create item data file
prepare execution of item
check time
execute
calculate used time
want item results (out: results)
extract alternative and score from item result file
erase non-influencing subjects from influencing list
log (in: out-stream)
write item model, attributes, parameters, result, time
show selected item (in: out-stream)
write item overview for user
show finished item
write item & result overview for user

The actual execution of items, being a platform-dependent function in need of external files, is included in the file manager.
Creation of the item data file is actually also a possibly variable function; now the requirement has been posed that the Items-D type interface must be used, but in the future perhaps another system will be used. A flexible approach is to have an external program convert the resources (parameters and answers) into an item data file. The resources are saved into a temporary resource file. By using the run environment as identification, it is even possible to use more interface definitions within one course.

Besides giving information about the subject for which an item was selected, answers may indicate awareness or unawareness of knowledge belonging to another subject. This sort of information is stored in the influencer. The influencer information can be interpreted as: if the application criterion applies (usually "the answer is equal to alternative \( x \)") then act as if a correct/incorrect answer was given to an item of which the subject and subject level parameters are given by the influencer. The required information for this is available in the item bank; influencers are read from the item bank when an item is selected.

**Influencer**

attributes

- subject pointer
- application criterion

services

- does apply (in: answer model; out: applies?)
  
  *returns if the influencer applies with this answer*

- get subject (out: subject)
  
  *return subject*

**Answer model**

attributes

- answer (alternative/class)
- score
- time used

**Item model**

attributes

- item representation in subject model

services

- fulfills criteria (in: criterion; out: desirability of item)
  
  *return a desirability index with respect to the criterion*

- want criterion value (out: criterion value)
  
  *return the value with which the criterion is compared*

- read (in: in-stream)
  
  *read item representation*

- log (in: out-stream)
  
  *write item representation*

- show (in: out-stream)
  
  *write item representation to user*
4.5.3 The file manager

Many actions as desired by the file manager all show the same structure. The fact that the "want to read" and "want to write" commands are repeated for several objects gives rise to the thought as to whether a generalised class could be made: a class knowing how to read and write from and to a stream. All the objects performing I/O inherit the I/O services from this class. The file manager can then ask an object to read or write, without having to know exactly what that object represents. However, sometimes not just one object must be read; the item selection process is an example of when more objects are read at once, and when the position of the data to be read by the object is not known beforehand. Therefore the original version of returning a stream is maintained; the services want to read and want to write are parameterised. The parameters to be used are course, steering data, item bank (all read only), item manager data, student model (read/write) and log and resource (write only).

Another task of the file manager is hiding the (domain-dependent and perhaps application-dependent) file structure. For fairly small data bases, it is convenient if all read-only information is clustered in the same file. Sections can be separated by keywords, comparable to the MS-Windows initialisation files. However, it is also possible that several courses use the same item bank, for example when there are several versions of a course for different studies. Knowledge about the file structure is available in the file manager. As said at the item manager description, yet another task is the execution of external programs. This is also platform-dependent; for example, under MS-DOS a swapping procedure is needed to be able to run large programs due to the limited addressability.

Attributes of the file manager are the directory in which the read-only resources are stored, and the directory where the session results have to be stored, and possibly read a next time. Because more sessions can be active at the same time, the sessions (or, more correctly, the students doing the sessions) will have to be identified. The student's identification number can be used, or, if more security is desired, a function of this number. The student number is delivered from outside. The two directories are read from an initialisation file when the object is created.

File manager
attributes resource directory
result directory
session id

services want list of courses (out: list of course names)
investigate which courses are available
A general computer-based training shell

put the names into a list
return the list
use course (in: course name, student identification)
convert student identification into session identification
identify corresponding resource and result directories
want to read (in: in-stream identification; out: input stream)
return an open input stream for the purpose identified
want to write (in: out-stream identification; out: output stream)
return an open output stream for the purpose specified
done reading/writing
if an input/output stream is open, close it
execute (in: item run parameters)
append resource directory and session id to parameters
execute run script with parameters

4.5.4 The user interface

In the previous sections, several components of the user interface have been identified. A text-based interface is shown in figure 4.5. All arrangements such as drawing boxes, reverse video and positioning on the screen are handled by the user interface itself. Filling the boxes is delegated to the corresponding objects, which use only standard I/O functions. For display of item information, a vertical format

```
last item description + result

student id

student model overview

student input (menus, questions)

system messages
```

Figure 4.5. Components of the user interface

—every piece of information being on a separate line—is most convenient. For the student model, each subject model will be allowed to use one line. It must be clear to the user what information is provided, but the information must also be compact. A table format with a header is a solution. Ergo, the objects will also have to provide
the header information. As the headers can have a different look, meaning that headers and data must be separated, the corresponding services will return two streams: one for the actual data and one for the headers. The three stream returning services can be combined into one general service with a display ID to discriminate among the different actions.

In critical situations (when an anomaly occurs, such as that execution failed, the result could not be found, there are no more items available), system messages, giving a description of the problem, can be shown to the user. Further, if an action occurs which could take some time (for example, retrieving previous session results from a file server), the user should be informed, to prevent the impression that the system hangs. These messages can be sent to the user interface as well. This implies that either the user interface must be visible for all other objects, besides the control object, or messages must be returned by the object’s services for processing by the control object. The latter is quite awkward because each service which could possibly generate a message must have an extra return parameter; this does not improve clarity. However, when making the user interface visible globally, too much information can be reached by the objects. To prevent this, an intermediate "message dispatcher" is needed, hiding the actual user interface.

A final consideration concerns the student identification. This must become known (by asking the user) before the "steady state" user interface can be constructed. A suitable time to ask is at the creation of the user interface object. The student identification is also needed in the file manager, in order to find the session identification. The two subjects communicate indirectly via the control object, which, at initialisation, first asks the user interface to pass the identification, and then gives it to the file manager.

**User interface**

attributes

| student id |

services

| want to show (in: action ID, out: header stream, info stream) |
|---|---|
| prepare presentation using action ID |
| return streams |
| show message (in: message) |
| if message concerns fatal error |
| - indicate severity |
| display message in system message area |
| want student identification (out: student identification) |
| return student identification |
| process menu (in: menu items; out: choice) |
| show menu containing menu items |
| repeat |
A general computer-based training shell

- get user input
- process user input
  until user makes a choice
  return item corresponding to choice
  ask question (in: question, max. answer size; out: answer text)
  show question
  retrieve answer
  return answer

initialisation ask student id

Message dispatcher
services show message (in: message)
  send message to user interface
  if message concerns fatal error
  - end program

4.5.5 The control object

The basis for the system is a simple blackboard architecture. The blackboard consists of several sub-blackboards. The sub-blackboards are more than just shared data; communication with them takes place only via messages and they keep their data valid themselves. There are three sub-blackboards:

- the Control blackboard, containing progress information for use in the blackboard controller.

- the Model blackboard, containing the structure of the course in terms of subjects, and a student model which is a reflection of that (i.e., the course model object)

- the Item blackboard, containing information about the current item.

The applied inference mechanism is backward chaining. This is a difference between it and usual blackboard systems, because the usual approach is a combination of backward and forward chaining; at each step both methods are attempted. The forward chaining process is opportunistic; the underlying reasoning is "if something can be done which gives new information, just do it and see what happens". The inferences are not so complicated that such an approach is necessary. Preferably, the inference mechanism must be goal oriented, making backward chaining a better choice.

Knowledge sources (KS) can be hooked into the system by making them known on the control blackboard. Each knowledge source contains preconditions, i.e., neces-
sary conditions before activation of the KS is useful, and possible postconditions, i.e., intended goals of the KS.

The control blackboard contains two lists, a *wanted* list with goals, and an *available* list containing already achieved subgoals. The knowledge sources are allowed to specify goals as being available or unavailable, and are thus able to influence the inference process. Making goals unavailable allows temporal reasoning; this is necessary in a continuous process because knowledge may vary in time. The parts of the control object are shown in figure 4.6.

![Diagram of control object](image)

*Figure 4.6. The control object*

The inference mechanism is started when a final goal is specified; it ends when the final goal has been reached. An inference loop starts with examining all KSs on usability. A KS is set executable when the postconditions contain a goal (i.e., one of the postconditions is wanted) and all preconditions are met (i.e., all preconditions are available). If preconditions of a KS are not met while the postconditions contain a goal, those preconditions are put on the goal list (made wanted). Now, the executable KS most probably leading to the goal is activated, and the inference loop is restarted if the final goal has not been reached yet. If there is no executable KS but the control blackboard has changed during the loop, a KS may be selected the next time, so the inference mechanism restarts. The mechanism fails if no change on the control blackboard was seen during a complete loop. In normal circumstances this does not occur; it indicates an error or incompleteness in the knowledge sources, causing the goal to be unreachable.

**Control mechanism**

<table>
<thead>
<tr>
<th>services</th>
<th>set wanted (in: goal)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>report that goal is wanted</em></td>
</tr>
<tr>
<td></td>
<td>set available (in: goal)</td>
</tr>
<tr>
<td></td>
<td><em>report that goal has been reached</em></td>
</tr>
<tr>
<td></td>
<td>set not wanted (in: goal)</td>
</tr>
</tbody>
</table>
A general computer-based training shell

report that goal is no longer wanted
set unavailable (in: goal)
report that goal is no longer available
is available (in: goal; out: available?)
returns true if goal is available
is wanted (in: goal; out: wanted?)
returns true if goal is wanted
add knowledge source (in: knowledge source)
add knowledge source to blackboard
infer (in: goal)
start the inference mechanism to find goal

The knowledge source is shown in figure 4.7.

![Figure 4.7. The knowledge source](image)

**Knowledge source**
- **attributes**
  - knowledge pointer
  - executable
- **services**
  - is executable
    - return true if KS is executable
  - set executable
    - make KS executable
  - execute
    - execute KS

**4.5.6 The list**

Several objects can occur on multiple occasions. To represent the multiple presence of an object, the list is used. A generic list specification is made, in which entry denotes an instance of the data of interest. The list does not allow duplicates; the addition of a duplicate and deletion of a nonexistent entry are ignored.
4.6 Choice of programming language

As a programming language, C++ has been chosen. This is a widely available, object-oriented language, but it allows also hybrid programs, i.e. programs using both objects and functions. An additional advantage is that it is possible to program at a low level without losing platform independence, by using standard libraries. For example, for all platforms, file streams with formatting services are provided, which form an appropriate implementation of the streams needed to display information.

4.7 Comparison with adaptive methods in control theory

It is interesting to investigate whether the adaptation process can be matched with an adaptive control method as known in control theory. Figure 4.8 shows a block diagram of the designed system in an adaptive system controlled situation. The resemblance with a self-tuning controller is striking. The inner control loop is
formed by the subject level, the outer loop by the course level. If the diagnostic unit concludes that a system parameter (the system being the student) has changed, it reacts by modifying the goal. An extra complexity is that the system output—the student's proficiency—is not observable; only responses to items are. From that, the proficiency has to be estimated as well.

4.8 Conclusion

A generally applicable adaptive training shell has been designed. The word "shell" implies that the system provides required functions, but is still empty. Objects to be elaborated are the course structure, the desired course adaptation mechanism, the subject level model and adaptation process, the control strategy (specifying the amount of student control), the user interface and file specific operations. The external behaviour of all these objects is known, but implementation is unspecified.

As the execution of items is left outside the system, items can be built in any environment that uses the same platform as the system instantiation.
An instantiation of the shell

The design of a general training shell was described in Chapter 4. This chapter deals with an instantiation of that shell. First the purpose and requirements are described. Then the choices regarding the models are discussed, followed by the control mechanism. Given the target operating system and the intended use, there are demands on the user interface and the data base management. Therefore these components are discussed as well. Then some functions that are handled in external programs, to support platform independency, are described. The chapter ends with several concluding remarks.

5.1 Purpose and requirements

The purpose of this instantiation is an adaptive session manager for an introductory course in electronics. The target platform is MS-DOS; items with an Items-D interface are available. The item difficulty level is roughly available. The item bank consists of multiple choice items and, in the future, probably multiple classification items. A multiple classification item is an item in which the given answer (most often numerical) is translated into an answer class. In the most simple case the classes are "correct" or "incorrect", but it can also have a more diagnostic character, for example, "knowledge of subject Y is used incorrectly". The items themselves do not "know" this subject, but the class represents the teacher's interpretation of the answer. In both cases, the return value is a scalar value, which means that the item descriptions in the data base must include an interpretation of the information returned by the item.

The prerequisite subjects must all be available at a sufficient level before a subject can be chosen. The desired method of traversing the course is starting at the bottom, taking each subject to the level in which it is likely that there is sufficient proficiency for continuation. If all subjects have been processed, the course model is tra-
versed again to lift the level to one of strong confidence. This strategy introduces new subjects more quickly, which is desirable since one session should give an overall impression of performance on the course, and not just a precise estimation of performance on a few subjects.

With the item selection within a subject, a gradual increase in difficulty is desired. It is the course lecturer’s opinion that this method is a good way to give training for this course. Extreme jumps, especially from easy to very hard, can influence the student’s confidence and must therefore be avoided. Further, the item bank cannot be assumed to be large enough to neglect the possibility that an item will be selected twice. If possible, repeated selection of an item should be avoided. It is better to select a somewhat less optimal item that has not been answered yet, than an optimal item that has already been done.

In electronics, many items use electronic circuits. If possible, consecutive items should select items in which the same circuits are used.

Since the difficulty levels are only known roughly, the selection mechanism must not be too dependent on the accuracy of the item’s difficulty.

5.2 Choice of models

5.2.1 High-level or course model

Structure

The course structure can be modelled as a subject lattice with and-links only, since the prerequisite relations in the course model must all be available in order to make a subject eligible.

Adaptation process

The adaptation process relies on the statuses of each subject. The status per subject consists of values unlearned, perhaps acquired, probably acquired, and reliably strong, as suggested by Lesgold and his colleagues. The principles of this notation are as follows [Lesgold et al., 1989]:

The perhaps state indicates that the student has been observed to perform a target skill, but there is not enough evidence to conclude that the conditions and actions are known. This is an unstable state; either further correct performances will occur, which causes a movement to the probably state, or the correct performance shows to
be accidental, causing the status to move back to \textit{unlearned}. Repeated reliable performance will move the student from \textit{probably} to \textit{strong}.

The implementation of the state changes is left to the subject model; it is important for the course model that the states are known. These are the bases for the adaptation process. A priority scheme is used in the following way:

1. As said, \textit{perhaps} is an unstable state. So in the selection of curricular goals (subjects), subjects with this state have a high priority.

2. The next issue is the selection of subjects which have become available because of acquired prerequisites. This means inspection of prerequisite relations of \textit{unlearned} subjects. A subject is available if the status of all prerequisite subjects is at least \textit{probably}. This choice is a compromise between adaptation speed and accuracy; the state means that there is reason to assume that the student masters the subject. Taking into account that the training sessions must give an impression of the complete course, it is not desirable to stay at one subject until the performance is reliably \textit{strong}.

3. The final issue is the selection of subjects with status \textit{probably}, to increase the performance on those subjects to status \textit{strong}.

An option is to include stochastic elements in the scheme, by translating priorities to probabilities of usage, but chances are that this would give a very cluttered impression. It is likely that the student must be able to see some logic in the adaptation process to remain motivated. Therefore, the three selection criteria are handled sequentially, and if there are more subjects with the same status and priority, the earliest subject in the curricular order will define the subject to be chosen.

The result of this method is that to obtain status \textit{strong} for all subjects, all subjects must have been passed at least twice, one to move from \textit{unlearned} to \textit{probably}, and one to move from \textit{probably} to \textit{strong}. This result fits the desired strategy exactly.

\section*{5.2.2 Low-level or subject model}

\textit{Drawbacks of standard IRT}

Since a difficulty specification is available, item response theory (IRT) is in principle applicable. However, there are some problems to be addressed. Standard IRT violates the requirement that no extreme jumps are allowed. If the first item is simple and it is answered correctly, the most likely next item will be the most difficult available, because the likelihood function of a test of one item is equal to the ICC of the item, which does not have a finite maximum. This could be solved by assuming
a prior distribution, but then still the gradual change in difficulty has not been established. Also, since IRT is a testing technique, an underlying assumption is that the student's proficiency does not change. With instruction this assumption is violated because solutions and hints are given, intended to improve proficiency.

Further, IRT depends on the accuracy of the item parameters. For fairly accurate estimates of item parameters, with the 3-PL model, at least 1000 individuals have to be confronted with the item. With the 2-PL model this number is still 500; using the 1-PL model the minimum is 200 [Frick, 1990]. With a target population of about 200 students a year it could take several years before the necessary accuracy for item parameters has been reached.

The standard IRT stopping criterion is a threshold value for the information function. As seen in Chapter 2, this function is a measure of the estimation error. With this criterion, too many items are needed before an accurate decision can be made. To illustrate this, a simulation of IRT sessions was done, with fixed values for the discrimination value a and the guessing parameter c for all items. As many of the items are 6-choice items, the c-parameter was set to 0.1, somewhat below the probability of guessing correctly. It is not set exactly on that probability; because distracters in the answer possibilities usually cause lower probabilities of guessing correctly; lower ability students are more likely to “fall into a trap”. The simulations done were repeated sessions with “virtual students” with proficiencies of -2, 0 and 2, while items with a difficulty range between -6 and 6 are available. The results of the simulation are shown in figure 5.1.

![Figure 5.1. Minimal number of items for three information thresholds](image)

---

76
Normally, most items have discrimination values between 0.5 and 2; even if we take an information threshold of 5, implying a rather large standard error in proficiency of about 0.45, and all fairly high discriminating values of 2, still at least 12 items per subject are needed. If it is desired to lead the student through all subjects in one session, this is clearly too much, since an item will take at least three minutes, and on average about six minutes (this data has been based upon observations in examinations and trial sessions).

**IRT with cut-off proficiencies**

An important question is whether the stopping criterion is justifiable for training purposes. We are not specifically interested in the exact estimate of the student’s performance, but merely in the question of whether the performance can be assumed to be above a certain level. Therefore, a better alternative for the stopping criterion is that the proficiency has a certain probability of being higher than a specified value, instead of at a specified value. This can be implemented by demanding that a confidence interval for proficiency is located entirely above a cut-off value for proficiency. Suitable values for the cut-off values can be obtained, for example, by using the test characteristic curve as explained in Chapter 2. For the confidence interval, the variance of the proficiency estimate is needed. In Chapter 2 it was argued that if the number of items administered is large enough, the variance can be approximated by the reciprocal of the information function [Wainer, 1990].

Using this value from the beginning is not entirely reliable, because for small numbers of administered items the approximation is not accurate. Further, the information function varies as a function of proficiency. However, in practice it works quite well. The described method will allow the implementation of criteria for the status by using different cut-off proficiencies. Although eliminating one of the drawbacks, still the other IRT problems, especially the problem of large jumps and the assumption that items do not influence the proficiency, remain.

**A heuristic approach**

Teachers usually take a more heuristic approach. They do not know the exact values for item difficulties, and still they are able to classify the students. What kind of rules do the teachers use? To examine this, several teachers were interviewed. They usually start with a simple question. If that is going right, a gradual increase in difficulty is applied. This level of difficulty does not consist of a fixed number, but is defined relatively, e.g., “this item is somewhat more difficult”. After a few items, teachers usually have a fairly accurate impression of ability, in fuzzy terms.
It was shown that one of the drawbacks of IRT is the assumption that the proficiency does not change. In a training environment, the goal is the improvement of proficiency. This implies that previous performance should be forgotten more quickly, since the current value could be different. A moving average provides an implementation of a forgetting structure, using the average of the last \( n \) observations as estimate. The higher \( n \) is, the more conservative the estimate. An observation consists of a difficulty and the response (correct or incorrect). The first idea is to include the difficulty if the item is answered correctly, because the observation shows that the student can answer an item with that difficulty correctly, and to ignore it if the answer was incorrect. However, there is a drawback to this method. If, for one reason or another, the student gets an item which is easier than his estimate and he answers it correctly, the estimate will drop. If he answers it incorrectly, the estimate is not influenced. Both are highly undesirable effects. Therefore, another approach is taken.

When a new observation is available, the difficulty of the item is inserted into the array of estimates. If the answer is correct, the lowest value is forgotten, or else the highest. The behaviour for more difficult items remains the same; that for less difficult items is more acceptable, as the observation is ignored if the answer was correct (the observation gives information we already knew), and it is included if it is incorrect (it contains information contradicting the current belief). The process is clarified in figure 5.2, in which 3 observations are stored (difficulties taking values between 0 for extremely easy and 1 for extremely difficult). The obtained estimate is used by the status updating process.

\[
\begin{align*}
\text{Initial estimate:} & \quad 0.3 \quad 0.5 \quad 0.6 \quad \rightarrow \quad 0.47 \\
\text{item} & \quad 0.7 \\
\quad \rightarrow & \quad 0.3 \quad 0.5 \quad 0.6 \quad 0.7 \\
\text{incorrect} & \quad \rightarrow \quad 0.47 \\
\text{correct} & \quad \rightarrow \quad 0.6 \\
\text{item} & \quad 0.2 \\
\quad \rightarrow & \quad 0.2 \quad 0.3 \quad 0.5 \quad 0.6 \\
\text{incorrect} & \quad \rightarrow \quad 0.33 \\
\text{correct} & \quad \rightarrow \quad 0.47 \\
\text{item} & \quad 0.5 \\
\quad \rightarrow & \quad 0.3 \quad 0.5 \quad 0.5 \quad 0.6 \\
\text{incorrect} & \quad \rightarrow \quad 0.43 \\
\text{correct} & \quad \rightarrow \quad 0.53 \\
\end{align*}
\]

*Figure 5.2. Examples of the estimation process*
In practice, a trade-off has to be made between accuracy and adaptation speed. A memory of two observations appeared sufficient. With three, the adaptation is quite slow, because the system wants more "certainty" before it decides to update the state. With two, the process is much faster while still requiring at least two items with average difficulty above the desired cut-off to be answered correctly.

**Adaptation process for the heuristic approach**

To avoid large jumps, the adaptation process uses the teacher defined strategy. The difficulty level with the latest observation was the value previously thought to be the best level to try, so it is a defensible choice. If the latest answer was incorrect, a somewhat lower value is taken, and if it was correct, a higher value is taken. The step size must be not too large, but also not too small. Values between 0.15 and 0.2 have shown to be reasonable. With lower values, the adaptation is too slow, and with higher values the gaps between two consecutive items become too large. It is also thinkable to make the step size dependent on the item, the number of items asked, or on the performance, but this has not been done in order to keep the process simple.

The advantages of this process, compared to IRT, are that it avoids large jumps, it is easy to understand and it does not need the characteristics of all applicable items before a suitable item can be chosen. Further, it can handle changes in the student's proficiency during the session by means of the forgetting mechanism. Its drawbacks are that it does not use the information as efficiently as IRT, and it does not have an accuracy measure. However, if the estimate is above a desired cut-off level, the student has shown to be able to handle problems with difficulties around the cut-off level several times, the number depending on the size of the history window.

The ability to handle changes in proficiency is caused by the fact that the desired difficulty changes between two items do not decrease, as is the case in IRT.

**Item model preference criteria**

The resulting difficulty can be interpreted as a fuzzy value; deviations are allowed to a certain extent. The limits to difficulty are passed to the item manager, which is capable of selecting a set of items according to external requirements (subject, difficulty limits) and internal requirements, described in the next section. The models of the items in this set are presented to the subject model to give a judgement. The criterion used for this judgement is a function of the difference between item difficulty and desired difficulty.
5.2.3 Item model

Item-course model

Via the subject lattice, for each subject the prerequisite subjects are known. It is still possible, however, that items test different skills which seem to have no direct relation. Some errors can offer evidence for a lack of knowledge on a subject prerequisite to the course, which is usually not to be tested because it is assumed to be known. Other errors can cause doubt about previously established prerequisites. For example, consider an electronics item using figure 5.3. In the circuit, the transistor must be biased at a certain collector current. For this, the DC voltage \( V_1 \) must have a suitable value. The student is asked to determine this value. Assume that —based on numerical data not given here— the given answer is: \( V_1 = +5 \) volt. Disregarding the numerical value, the sign of the answer is incorrect. If \( V_1 > 0 \), then from the voltage law of Kirchhoff for the circuit \( V_{in}, R_2, V_{BE} \) of \( Q_1, R_1 \) and \( V_1 \) it can be deduced that \( V_{BE} < 0 \). Then, the base-emitter junction is blocked, which is in contradiction with the condition for biasing the transistor in the active area. This indicates that the student probably has not understood the transistor operation.

It can also be that an item provides evidence that another subject is known very well. Since the items are all multiple choice or multiple category items —for which the answers or categories will from now on be called alternatives—, it is possible to specify probably influencing subjects per alternative. The amount of influence of the (lack of) knowledge of that subject is given by a item-subject model, used to update the corresponding subject model. For example, a low difficulty value can indicate that a very elementary mistake has been made in such an influencing subject. It can be seen as the statement “consider this mistake as if an item on that subject with difficulty \( b \) has been answered incorrectly”.

Figure 5.3. Circuit with an example item
5.3 The control mechanism

Item-subject model

The item-subject model specifies the information needed to update the corresponding subject model. As described in section 5.2.2, this model needs the item’s difficulty value.

Item selection strategy

The strategy is based upon the subject as indicated by the course model and the difficulty as indicated by the subject model. Items are chosen with a difficulty “close enough” to the desired difficulty. The definition of “close enough” is given by the subject model. To each selected item, a tag is connected indicating if it has already been answered during the session, and if the item has the same item series identification (if any). Using these tags, a priority scheme was developed:

- Top priority items are those which have not been answered yet and fall within the current item series.
- Second in the list are items which have not been done but have a different item series.
- Items are used more than once only if there is no other possibility. A parameter variation can still give a slightly different appearance.

With this priority scheme, items within the same series are selected first, but not at the expense of asking already answered questions. Further, items within the same series for which the difficulty is too far away are not used.

If after this still more items are eligible, the subject model can to a certain extent decide which one to choose. The subject model can give judgements about the possible items by returning criterion values; the item selection mechanism then groups items with close criterion values and randomly selects one.

5.3 The control mechanism

It has probably become clear already that the control strategy is adaptive program control. Filling in the control structure, six knowledge sources can be specified: course selection, subject selection, item selection, item presentation, model maintenance, and student communication.

The goals used at the control blackboard are course, subject, item, answer, student model, communicate, and the main goal finish. This section describes the knowl-
edge sources, their task, and their activation records, in the form of pre- and post-
conditions. For easy recognition in the descriptions, goals are printed in emphasized
style.

Course selection

The course selection represents the initialisation source for the reasoning process, it
must be the first one executed. It retrieves a list of courses from the file manager,
passes this to the user interface to have the student select one of them and tells the
file manager to use the selected course. Then the user interface is asked for a student
identification, which is passed to the file manager to transform into a session identi-
fication, enabling the file manager to locate possible previous session results. There
are no preconditions, the postcondition is the availability of a course.

Subject selection

The precondition for subject selection is that a student model and a course are avail-
able. Naturally, the postcondition normally is that a subject is available. This source
uses the course model to determine a subject. If it cannot find one, it sets the final
goal finish available, because it is not useful to continue if no subject can be found.

Item selection

The preconditions is the availability of a subject, the postcondition is availability of
an item. Further, communication is allowed after selection of an item. This source
uses the subject model to obtain a criterion for item selection, and the item manager
to select a suitable item. After selecting, it displays information about the chosen
item to the student. If the source is not able to select one, it invalidates subject. A
side effect of the item selection is that the student model is invalidated if an item can
be found; with the new item, the model is no longer up to date.

Item presentation

The item presenter can be executed if an answer is desired, provided that an item is
available. It uses the item model’s execute service to run the item. Afterwards, the
answer is available. By answering the item, it becomes unavailable (note that there
is a difference between the abstract notion item and the chosen item from the set).
5.4 The user interface

**Student model maintenance**

If the *answer* is available, the *student model* can be updated. This source uses the course model’s update method (which, in turn, uses one or more subject model update methods). When the student model has been updated, the *answer* is no longer available. The item result and the updated student model are shown to the user.

**Control communication with the student**

The student has no control influence, except from being able to stop this session. The student is given this possibility whenever the system thinks it is suitable, by setting the goal *communicate*. The postcondition is *finish*, which is the main goal for the inference mechanism. Therefore, if the preconditions of this source are met, the source will always be executed with priority.

**Conclusion**

Although the knowledge source descriptions translate quite easily into pre- and postconditions, some considerations must be made: sometimes it is necessary to steer the mechanism by explicitly invalidating reached goals. For example, to make the system execute the found item, the goal *answer* must be wanted, and for that, the goal *student model* must be wanted. Thus, while, strictly speaking, the model is valid until an answer is available, it has to be invalidated at the moment that an item has been chosen. This is due to the reasoning process of only backward chaining. The problem is not so severe that it would justify inclusion of forward chaining as well.

5.4 The user interface

The target platform being MS-DOS, the PC-ASCII character set is used for the creation of borders. The PC-specific *constream* object is used to obtain nonstandard console functions to position the cursor, change colour, and restrict access to a specific part of the screen. Constream is a child of the standard stream class (providing standard I/O), so all standard stream services are present as well. The preparation phase for the display services of the item and the course model consists of drawing a border, restricting the access of two streams to a specific part inside that border, setting their colours, and returning the streams.
An instantiation of the shell

The menu selection must avoid menus with only one selection; in that case the only menu item is the result. The implementation is as follows:

if only one menu item
    return item
restrict access to the student input part
make the first menu item active
repeat
    display all items with the active one reversed
    read user input
    switch user input:
        case arrow left: make the left neighbour active
        case arrow right: make the right neighbour active
    if no active item could be found, go to the other side
until user input is return
return the active menu item

The display of error messages is implemented as:

restrict access to system message area
reverse video
if fatal error
    set background colour to fatal error colour
display message

Asking a question becomes:

restrict access to student input part
write question
restrict access to field of max. input length
read string
return string

5.5 The data bases

The basic assumption is that the current directory is used to store intermediate results. It can be the student’s home directory on a server, or a temporary one. More permanent results are stored in a separate result directory. This allows server systems to gather the results in one server directory; students can use any client PC they want for a next session. Items, course structure and other resources which are read-only for the student are put into a resource directory. Therefore three directories are important: the current directory, the resource directory, and the result directory.
When asked for a list of subjects, the first action of the file manager is to investigate where this data can be found. Courses for which training material is available are mentioned in an initialisation file which is located in the directory in which the program executable resides. The course can be specified as a command line parameter; in that case it is checked if the name occurs in the initialisation file, and if it does, the name is returned. If there is no valid command line parameter, the file manager extracts the course names from the initialisation file into a list and returns that.

When informed of a course to be used, the file manager extracts the corresponding resource directory from the initialisation file. In the resource directory, a course description file is located, giving information about the result directory.

This course description file contains all read-only data needed for the course, except the item executables and data files needed during execution. Besides the location of the result directory, it contains sections for the description of the course structure, the item bank, the default student model, and the student model steering parameters. A section start is denoted with an identification word in brackets.

For a read request for read-only data, the procedure becomes:

`assign a stream to the course description file`  
`open it for reading`  
`go to the start of the specified section`  
`return the stream`

For a read request for read/write data:

`get file name belonging to data`  
`assign stream to file`  
`open it for reading`  
`return stream`

In connection with writing there are some considerations. There are two types of writing, one is appending data (logging), and one is overwriting data (saving a new version). When overwriting, history is lost. To preserve a history, a new version can also be inserted at the beginning. For the object in question, this change is not relevant since it reads one version, but for off-line analysis purposes it can be valuable.

For a write request for read/write data:

`get file name belonging to data`  
`assign stream to specified file`  
`open it for insertion at beginning`  
`return stream`
An instantiation of the shell

For a write request for write-only data:

- get file name belonging to data
- assign stream to specified file
- open it for appending
- return stream

A "done writing" message will cause the corresponding stream to be closed.

5.6 External programs

The session manager uses several externally made programs. The main reason for this is platform and item independence; for example, to use a new item interface, two simple dedicated programs have to be rewritten instead of having to change and recompile the session manager. This section briefly describes the functions of the needed programs and the current implementation.

5.6.1 Item interface programs

Data file creator

The format of Items-D data files is used to communicate with the items. A template for an item data file is available; this is a file with the correct structure where only the values of parameters have to be filled in. The program copies the template to the actual data file name, fills in the parameters and answers and closes the new file. At the moment the implementation is quite rigid, allowing only a pre-specified number of parameters and a specific answer, but this can be extended easily to become more tolerant.

Result extractor

The item result file contains much information not needed in the session manager. The result extractor searches for keywords in the item result file and saves the values corresponding to score and answer in the result specification file, for later reading by the session manager.
5.6.2 Data file mover

The data file mover is responsible for retrieving the needed data from the result directory into the current directory and storing new data in the result directory. Parameters are the command (get or put), the result directory, and the session id. Using this file, no platform-dependent constructs will have to be incorporated in the session manager.

5.6.3 Run script

The run script actually executes the item run environment with the desired item. Parameters to the script are the item identification, an environment identification, the session id (these three are usually file names without extension) and the resource directory.

The run script can be divided into two parts: the environment-dependent part and the environment-independent part. For every environment, a script is written with the environment specifics, i.e. how the environment executes items, what data files are needed, etc. The parameters for this script are the item name and the resource directory. The name of the environment-specific script is equal to the environment identification, allowing the item-independent part to execute it without case statements.

The independent part extracts (and if needed decompresses) all files starting with the item identification from the resource directory; these are the files needed for execution of the item. It also takes care of calling the item interface programs; this is done for the practical reason that then the session manager will only be swapped out of memory once. The actual interface files are logged to an interface log file with the session id as file name and a special extension. Further, exception actions can be specified: what to do if the item cannot be found or if another problem occurs. At the moment, such a calamity is logged to a problem log file. When the item cannot be found, a default item is fetched instead, reporting the problem to the student. As such errors should not occur (they indicate corrupted data bases!), the student is advised to inform the person responsible for the training sessions.

5.6.4 Executing external software under MS-DOS

Executing external programs needs some special consideration under MS-DOS. The amount of available memory is limited. Especially if the machine uses network software or other resident programs, it is often not possible to have both the session
manager and the called program in main (conventional) memory. Therefore a swapper is applied. A swapper moves the calling program into another type of memory (e.g., extended memory or hard disk) together with needed status information, then loads the called program over the calling program and executes it. After execution, the calling program is restored and the status is recovered, giving the impression that nothing has changed since the call command.

5.7 Conclusion

An instantiation of the general training shell has been realised for a course in electronics, under the operating system MS-DOS. In fact, the teacher’s preferred strategy was implemented: a subject cannot be chosen before all prerequisite subjects have at least a level showing some evidence of mastery, and if all subjects have that status, it is attempted to increase the levels to show strong evidence of mastery. There is a gradual change in item difficulty levels.

The requirements resulted in a choice of not to use Item Response Theory as such, and to implement a heuristic subject level model. The item parameters are comparable to IRT, but the update process is different. As finish-criteria, cut-off values for mastery level estimates are taken. Simulations have shown that this model provides fairly accurate values for the estimation process, and it implements the teacher strategy well.

It is well possible to fill the data bases with other course information and items. The system is usable for various courses without changing the program, but for major changes to the training strategy, programming experience is needed.

Tests have shown that the method of using external programs for data file creation, data file interpretation and item execution works well. In Chapter 7, a system test involving students is described; it investigates whether the students and the teacher appreciate the system, and if the results are reasonable. First, Chapter 6 describes the item implementation environments used and the needed extensions. The chapter is focused on a course in control engineering; this is the course for which the first non-adaptive training system was built, using Items-D as the session manager.
Environments and tools for item implementation

In the previous chapters, the management system was described. This chapter first briefly describes the courses piloted for this project. Next, the used item implementation environments will be described, followed by a description of the needed extensions and tools. Then an item template using an external simulation package will be described. The chapter ends with several conclusions.

6.1 The pilot courses

A system as described in this thesis can be used best with courses that are followed by a large number of students. With small groups, the effort put into the implementation of items and maintenance of the system is often higher than with classical training sessions. Within Electrical Engineering, there are a number of courses that are followed by a large number of students, especially in the first and second undergraduate years.

The project was initiated by the lecturer of the course “System and Control Engineering” and also the lecturer of the course “Electronics” showed interest in the project. Both had similar problems: they serve a large number of students, so a lot of time is taken up in instruction, especially in problem-solving activities. With the current decrease in manpower, it all came down to a very small number of instructors. Further, the instructors noticed that there are characteristic problems the students have with the courses, which return every year.

These courses are described in the following sections.
6.1.1 The course "Systems and Control Engineering"

For several years, the Control Laboratory has taught an introductory course on systems and control engineering to undergraduate electrical engineering students. The course is the basis for more advanced control courses taught by the Control Laboratory. The aim of the course is to let the student discover the structure of systems and the effect of dynamic properties in relation to the design of control systems, by means of several theoretical and practical control problem approaches.

From the Laplace transformation, the representation of dynamic systems is handled after which several fundamental system properties are covered, such as stability, sensitivity and optimality. For single-feedback systems the design procedure is covered to achieve a desired system behaviour (e.g., using the root locus method). Further, the course covers the frequency approach and the extended Nyquist stability criterion. In a second part of this course, analysis and synthesis procedures in the time domain are presented, both for continuous and for discrete systems. This includes topics such as state feedback and pole placement. The course ends with a short introduction to computer-controlled systems.

The written information for this course is contained in the book "Modern Control Systems" [Dorf, 1989] and a syllabus [Honderd, 1989] with additions, some clarifications, and elaborated problems. The course is divided as follows:

- teaching theory
- weekly assignments in the form of handouts issued during the teaching period
- problem solving
- obligatory laboratory
- voluntary CAD/CAL practice.

The system-oriented character of control theory is observed as a common difficulty, where nearly all preceding courses in the undergraduate curriculum have stressed the analysis of systems. Here the aim is the design and synthesis of certain control algorithms, given more or less known process behaviour. This demands a thorough understanding of the criteria for desired system behaviour. Also insight into the dynamic behaviour of systems and the different representations is needed. Further, the students need to obtain insight into sensitivity analysis and the effects of feedback and feed-forward control structures. Therefore, in addition to the lectures, instruction is carried out in problem solving and related activities. The emphasis of the instruction is on the enhancement of understanding and the comprehension of concepts involved rather than on calculation exercises.
6.1.2 The course "Electronics"

The Electronics Laboratory is responsible for the first-year undergraduate course in electronics. The course focuses on analog electronics, i.e. electronics for the processing of analog signals. Although it sometimes seems that everything is becoming digital, there is still an important place for analog techniques. The reason for this is that the world around us is analog, and the information extracted from it is in the first instance an analog signal, which, in addition, is sometimes a weak signal that has to be amplified before being processed further. The function of amplification is essential for the electronic transfer of information; therefore the electronic amplifier is the main focus of attention in the course.

Starting with conducting mechanisms in solid materials, some types of semiconductor components (diode, transistors) are treated. To facilitate computation in circuits with these components, circuit models from these components are formulated. From a network-theoretical point of view, a design concept is presented for the design of amplifiers with an optimal transfer and realised with components with limited precision. As applications, several amplifier configurations are discussed, with an operational amplifier and with a few transistors as amplifying elements. Finally, an introduction to current manufacturing techniques (the chip) is given.

The aim of the course is to acquire knowledge and skills regarding:

- the aim and function of electronics, the possibilities and constraints
- the operation and characteristics of diodes and transistors
- the analysis methods of electronic circuits (modelling, computations on electrical circuits)
- the transistor as amplifying device
- design criteria for amplifiers
- a design concept for amplifiers and application of the concept
- manufacturing techniques.

The literature for this course can be found in the book "Inleiding in de Electronica" [Wissenburgh, 1991] together with a collection of problems. The course consists of teaching theory and instruction sessions with emphasis on problem-solving. Some experiments with practical circuits are included in the instruction, particularly to

---

1. translation: Introduction to Electronics
support understanding of the relation between biasing and signal behaviour of non-linear components.

It is considered important that the students are actively involved at an early stage. Especially the understanding of a schema of an electronically device as an electrical circuit needs much training.

6.2 Requirements on the environments

When choosing a suitable environment for the development of items, many aspects should be taken into consideration. These requirements can be divided into two categories: requirements which an environment will have to meet in order to be able to be applied in the system, and requirements depending on the aims of the project. They will be called hard requirements and soft requirements, respectively.

6.2.1 Hard requirements

Items must be able to communicate with the management system. As described in section 4.4.3, this takes place by means of data files. By consequence, the environment system must support reading and creating of data files with a specified format.

Further, the system structure requires that the items can be executed on a stand-alone basis. This means that the items must be executable, or there must exist a way of executing the item and returning to the operating system without the user having to or even being able to interfere. This is necessary for two reasons: first of all we want the students to be able to concentrate on the items, and not on irrelevant actions they might have to perform to be able to activate an item. Second, we want to be certain that the student has done the item he or she was expected to do. If it is possible to start another item instead of the desired one, the outcomes of the sessions are not reliable.

6.2.2 Soft requirements

One of the most important aims is that the system should be easy to maintain. Working with the environment should be easy to learn and to memorise.

The emphasis of the training sessions for which the system will be used is preparation for the examination. In both pilot courses, graphics are used intensively. In control engineering one can think of block diagrams, pole-zero configurations, Mason-type flow diagrams, various types of time- and frequency responses and schematics
of systems. In Electronics, schemes of electrical circuits and frequency responses are used in particular. With ease of use for the teacher in mind, it should be easy to generate these kinds of graphics. Further, it must be possible to display formulae.

Many items can be generated with only the requirements above in mind. Training sessions for a pen and paper exam using only pre-calculated graphics were found to be useful to students (see Chapter 7). But using a computer we can do much more than just computerising the paper items. There is a large potential in computing power. We can do things that are impossible in class on the blackboard, such as online simulation and complex calculations. For example, it is very clarifying for a student to show the influence of parameters interactively. This means that the student can enter any value for certain parameters and have the system show the result. In such cases, pre-calculation is no longer feasible. For many applications, useful software is available. The application of existing special purpose software is to be preferred above “re-inventing the wheel”. A method would be to offer the special purpose software as a tool. Unfortunately, this software is often not easy to work with for beginners. Because the primary objective of the system is to give training in problem solving, and not to teach how to use a certain program, it would be better to shield the students from complicated software.

6.3 Description of implementation environments used

As a rule of thumb, one can say that for static items (i.e. not requiring dynamic complex calculations) authoring systems can give satisfactory results. If a more dynamic behaviour is desired, and more sophisticated functions are needed, it is usually better, if not necessary, to use a general programming language. There is also a hybrid option. If an authoring system allows addition of new functionality, the standard part of an item can be implemented in the authoring system, and the more complex part in a programming language.

For the pilot projects on control engineering and electronics, most experiments were done with the MS-DOS based authoring system Taiga. This was done for practical reasons; at the start of the research Taiga was the only available authoring system with which people had experience. Further, the programming language Turbo Pascal was used. As the authoring system is probably fairly unknown outside the Netherlands, a short characterisation is given in the following section.
6.3.1 Taiga

Characterisation

The acronym Taiga stands for Twente Advanced Interactive Graphic Authoring system [Taiga, 1987]. It is an MS-DOS based system, designed to allow non-programmers to produce educational software. Taiga consists of two parts: the production part, in which the course applications are built, and the distribution part, which runs the applications.

A Taiga application is organised like a book. It consists of one or more modules. A module is divided into paragraphs (sections), which in turn contain episodes. The episodes consist of frames, which can be classified into five categories:

- Presentation frames show or erase texts, numbers or characters. Examples are "display this text on screen" (Text), "show this graph" (Graph), "show the contents of a variable" (Output), "clear (a part of) the screen" (Erase).

- Reaction frames enable the student to enter a reaction and process the reaction. There are three reaction frames: "get input from the student" (Input), "wait for a few seconds" (Wait), "process a multiple choice question" (Multc).

- Control frames define the path in a CAI course. For example: "assign a value" (Assign), "Go to another point in the episode" (Jump), "compare a variable" (Check).

- Registration frames allow the CAI programmer to document his program and to follow the progress of the student during the problem.

- An external call frame allows external programs, written in Microsoft Pascal, to be executed (Progr). To improve readability of an episode, the programmer can assign a label of, at most, 8 characters to these external programs.

Figure 6.1 shows a screen capture of the episode editor. It contains a simple example of the flow of an episode. The execution starts above left. First, the question text is shown (Text 01). Then the frame Input CA waits for user input and assigns this to variable CA. The Check CA checks if the answer is correct. If so, execution resumes after the frame, else under it. In case the answer is correct, a feedback is shown (Feedb 04), the program waits for 5 seconds (Wait 05) and execution is terminated (because no next frame is specified). If it is incorrect, a feedback is given (Feedb 10), the program waits for a key press (Wait 00), the feedback is removed (Erase PP) and the question is asked again by jumping to the second frame (Jump 02).
6.3 Description of implementation environments used

![Diagram showing a simple Taiga episode]

Figure 6.1. A simple Taiga episode

Advantages and disadvantages

The main advantages of Taiga are:

- it is very easy to learn and remember.
- it is possible to use templates, so it is easy to make new items once a template library has been established.
- Taiga supports many facilities for dealing with various question types, such as numerical, textual, dichotomous and multiple choice questions.
- It is possible to extend Taiga’s possibilities by linking code.

Disadvantages are:

- it is a bit unclear where to find what if the items are not structured well.
- the text editor is rather obsolete (especially when entering formulae or repositioning text), and many Greek symbols are unavailable.
- text is often scattered though various frames so it is not obvious at a glance what will be on the screen.
- it is impossible to execute external software which cannot be linked to Taiga.
- graphics facilities are not very sophisticated.
- use of files is not supported.
The first disadvantage can be solved by structuring the items properly; the last two can be solved by implementing extra functionality. The special symbols can be made with a special font-editor; there are two alternative sets of 256 characters available for redefinition. Comparing the advantages and the disadvantages, it seemed a good solution to implement the extra functionality for Taiga and to use Taiga for the more static items. The requirements and the design of the Taiga extensions are described in sections 6.4 to 6.7.

6.3.2 General programming languages

Although authoring systems have the advantage of being easy to learn and being more accessible for non-programmers, they have limitations. For some special purpose items, it can be very recommendable to use a general programming language. A programming language offers more flexibility, but it usually takes more time to realise a program. On the other hand, languages like Turbo Pascal and Borland C++ offer many advanced features, reducing the needed implementation effort. For example, a library can be created for communication with the management system. This means that it only has to be implemented once and that, from then on, it can be used whenever applicable.

As these programming languages are well-known and the subject of many books, they will not be described any further here.

6.4 Taiga extensions: software requirements specification

6.4.1 Introduction

The basic Taiga system is very closed; everything must be implemented within the production system. For application as an item implementation tool, Taiga must be able to interface with the session manager. For this purpose, an extension will have to be realised. Often figures are available for numerous items, because they have already been made for exams or exercises; a facility to be able to use these would save much time to the item implementer. Another welcome extension would be the ability to display externally prepared texts. This would enable item implementers to make libraries of texts. Two major advantages of such an extension are:

- If the extension allows more than one section of text per file, all texts of an item can be grouped together in one file, giving the implementer a better overview of the texts in the item.
6.4 Taiga extensions: software requirements specification

- The Taiga programs can be used as templates for a large number of items. The item implementer would need only one Taiga program for each item type. Therefore, attention can be focused on the item content in stead of implementation activities.

More specifically, in control engineering many graph types are used, for system specification and for the description of system behaviour [Essenius et al., 1993]. It would be very convenient for the item author if it were possible to show such graphs in the items. Often those graphs are the results of applying simulation software, and not of using a general drawing program. Many of those graphs can be marginally changed to obtain different versions of items. However, if the graphs would be made with the Taiga facilities or even with the just discussed image drawing utility, it would be impossible to make changes easily. For example, figure 6.2 shows three possible pole-zero configurations of a very simple system with one pole (denoted by a cross) and one zero (denoted by a circle). The position of a pole can have a random value between -1 and -4, the zero has a value of -2.

![Figure 6.2. Examples of pole-zero configurations with a varying pole](image)

Using higher order systems (which means more poles and possibly more zeroes), the number of possible configurations will increase drastically, even if an approximation of the correct location is allowed. It would be very inconvenient if for all the possibilities a graph has to be drawn beforehand. Therefore, it is desirable to have functions that are able to draw these plots from specifications.

In summary, the system will form an extension to the authoring system Taiga. It will allow teachers to use Taiga for implementation of items for a pre-examination training system as described in Chapters 4 and 5. Three different parts can be specified:

- An interface between the management system and Taiga, called Items-D type Interface for Taiga (I²T). Its purpose is to retrieve the necessary item information, to maintain the item status, and to return the resulting information to the management system.

- General extensions to enhance item implementer efficiency, called General Toolbox for Taiga (GT²). These consist of display facilities for externally created images and texts.
Specific extensions for electrical engineering and control theory problems, called Control Toolbox for Taiga (CT²). These consist of display facilities for various types of graphs often used in control theory.

An essential model of the product is shown in figure 6.3. The circles denote the extensions to be implemented. CT² and GT² are not handled separately because their connections are identical on this level.

![Diagram](image)

*Figure 6.3. Essential model of the product*

**Definitions, acronyms and abbreviations**

- **Authoring system**
  - A system for courseware implementation with emphasis on content instead of on programming.

- **CT²**
  - Control Toolbox for Taiga.

- **GT²**
  - General Toolbox for Taiga.

- **I²T**
  - Items-D type Interface for Taiga.

- **Item**
  - Implementation of a problem.

- **Items-D**
  - Interactive Test and Exercise Management System Delft. The system of which the interface specification with items is adopted.
6.4 Taiga extensions: software requirements specification

Management system  A system managing a data base of items; one part creates sessions for the student; the other part provides a production system for the teacher.

Subquestion  An item can contain more than one question. These are called subquestions, even if there is only one in the item.

Taiga  Twente Advanced Interactive Graphical Authoring system.

Bode diagram  A frequency response plot in which modulus and phase are plotted against logarithmic frequency.

Frequency response  The steady state response of a system when a sinusoidal signal is applied, for various frequencies.

Mason-type flow diagram  A representation of a system in which the connection between the quantities is given along directed lines, called edges. Signals are represented by vertices.

Nyquist diagram  A frequency response plot in which the real and imaginary part at a specific frequency are plotted against each other. A Nyquist diagram is also called a polar plot.

Pole-zero configuration  The graphical representation of a system in the Laplace domain, with poles and zeroes.

Time response  The response of a system in the time domain to a certain input signal.

6.4.2 General description

Product perspective

The product will be integrated with the Taiga run-time environment. This system runs on IBM PCs or compatibles with at least a VGA graphical card and the operating system MS-DOS. The product will be used as a library of procedures, callable when desired in a Taiga program.

The subproduct $I^2T$ will rely upon a session manager to provide the item information and to process the item results.

Product functions

The most important functions of $I^2T$ are to take care of retrieving item information consisting of parameters, correct answers and other score influencing factors, to update the status of an item when new information becomes available (e.g., calcu-
iating scores when an answer is given), and to return the final status to the session manager. Checking answers can be done within Taiga or can be left to I²T. As part of the interface specifies files to be available on certain occasions, I²T will take care of this too.

GT² will provide facilities for display of externally created texts and figures; CT² will take care of display of continuous and discrete time responses, Bode and Nyquist diagrams, pole-zero configurations and Mason-type flow diagrams.

**User characteristics**

There are two types of users. On one hand there are the item implementers, who are usually teachers without much programming experience, or their assistants. On the other hand there are the students who will have to solve the problems in the items, and have to deal with the graphs. Students are usually first-year undergraduate students in technical studies such as electrical engineering, mechanical engineering, computer science and physics.

**General constraints**

I²T will use the interface specification as used in Items-D; CT² will use the format as delivered by the package TRIP [van den Bosch & Butler, 1990] when appropriate. Concerning GT², the text display will accept at least normal ASCII files and the figure display will use a format for which the majority of drawing programs has an export utility. Taking into account that figures are used extensively in technical subjects, the format must also be compact.

The programming language to be used must support the Microsoft object format, to allow linking to Taiga. This means that a Microsoft compiler (C, Pascal or Fortran) must be used. In the Taiga manual and in the software, all examples and interface definitions are specified in Microsoft Pascal. For later maintainability, Microsoft Pascal is therefore recommended.

The design must be general enough to facilitate porting to other authoring systems without too many problems. More specifically, this means that interfaces with Taiga should be grouped and that care should be taken not to use too many Taiga specific features. This restriction does not apply to the General Toolbox, as this toolbox adds functionality that should have already been in Taiga, and it uses Taiga specific formats.
6.4 Taiga extensions: software requirements specification

Since the system is designed to be used for various courses, the three subparts must be made independent of each other, so it is possible to leave out any of the subparts when required.

The primary objective of CT$^2$ and GT$^2$ is to offer item developers a set of graphical tools which are easy to use. For all types of graphs in CT$^2$, it must be possible to specify the position, size and colour of graphs, and, if appropriate, the boundaries of the graph. Also, reasonable defaults should be available. Further, if axes are within the visible area, they should be shown.

Preferably, the drawing routines must require as little stored data as possible, to keep as much space as possible available for applications. Slower functions that use little memory are to be preferred above fast functions that use much memory, provided that the performance is acceptable.

6.4.3 Functional requirements of i$^2$T

Get item information

Purpose: Retrieve parameters, anticipated answers to subquestions and other information related to the item. Make the parameters accessible to Taiga.

Input description: The item information according to Items-D specifications, and possibly the item result file (e.g., if the item is executed a second time after quitting before answering completely).

Processing: Make a new internal structure for a temporary storage of the answer information to all subquestions and store the available answer information.

Output description: The parameters in Taiga variables and the internal structure.

Get information on subquestion

Purpose: Make information about a subquestion available to Taiga.

Input description: Taiga provides an identification for the desired subquestion. Further, the internal structure is used.

Output description: The status (finished/aborted/not done), the best answer, the number of attempts used, and the maximum achievable score are returned to Taiga.
Environments and tools for item implementation

Check answer

Purpose: Check a given answer to a subquestion, and update the score for that subquestion.

Input description: Taiga provides the subquestion identification and the given answer.

Processing: Compare the given answer to the answer alternatives and update the internal structure accordingly.

Output description: Returned to Taiga are an indication if the answer was correct, and if it is allowed to try again. Further it returns the remaining number of attempts, a line of feedback, if available, and the correct answer.

Register answer and score

Purpose: Enter an answer and its corresponding score, that has been checked in Taiga already.

Input description: The Taiga application provides the subquestion identification, the answer and the score.

Processing: The internal structure is updated with the information and the subquestion is marked as finished.

Register score influencing factors

Purpose: Register occurrences possibly influencing the score to be assigned, for instance if a hint was asked.

Input description: Taiga provides the subquestion identification and the occurrence identification.

Prepare data files

Purpose: The item information may contain prototypes of data files which must be available at a certain occasion, for instance, just prior to executing an external program. This function makes data files from such prototypes.
6.4 Taiga extensions: software requirements specification

**Input description:** The file prototype(s) and the Taiga variables used in the file prototype(s).

**Processing:** In the file prototypes, the representation of Taiga variables must be replaced by the content of that variable. In text variables, the character “l” will be replaced by a new line character.

**Output description:** The required data file(s).

*Make item result available*

**Purpose:** Make the item result available to the external world.

**Output specification:** The item result (i.e. a representation of the internal structure) according to Items-D specifications.

### 6.4.4 Functional requirements of GT²

**Display of external graphs**

**Function:** The display of an externally created image on a specified part of the screen. For the format the GIF 87a standard [CompuServe, 1990] is chosen. This is a well known bitmap format for which nearly all drawing programs have an export utility. The GIF-format is compact, caused by application of a data preserving compression algorithm.

**Input description:** From Taiga, a window is specified in which the image must be drawn. The coordinates are in Taiga GKS (Graphical Kernel System) units, in which the screen is divided in 3200 virtual dots wide and 2400 dots high. Further, Taiga specifies the file name of the image to show, and, in case of binary (2 colour) images, optionally two colour codes for the foreground and background colour. The last part of the input is the file corresponding to the given file name.

**Processing:** During decoding, colours will be matched as closely as possible to the available Taiga colours. If the image size does not match the desired size on the screen, the sizes are multiplied by the largest integer number using which the complete picture remains inside the external border. The aspect ratio of the image may not be altered. A multiplication of the picture size implies an enlargement of pixel size.
**Output description:** The image in a window on the screen. To allow backgrounds not to become erased, the colour black is not drawn in case of binary pictures. If the file cannot be found, the function returns an error status to Taiga.

**Display of external texts**

**Function:** The display on screen of a piece of text, possibly including changes of character sets and application of the attributes underline, reverse video and colours.

**Input description:** From Taiga, the top left position of the text is specified. This is done in text coordinates (80 wide, 25 high). Further, Taiga provides the page number of the text, and the name of the text file. The text file itself belongs to the input as well. The format of the text file is described in Appendix A.3.

**Output description:** The page of text on a specific position on the screen. If the file or the correct page number could not be found, a error status is returned to Taiga.

### 6.4.5 Functional requirements of CT²

**Functionality identical in all facilities of CT²**

**Function:** Display of a control theory related graph on the screen

**Input description:** Taiga provides a window in which the graph must be displayed in GKS coordinates, the file name of the response, the frame colour and the response colour.

**Output description:** A graphical representation of the desired input file in the desired form. If the file does not exist, or does not contain the information necessary to make the graph, control is returned to Taiga with an error status.

**Display of time responses**

**Function:** The display of a time response in a window on the screen. Depending on the input file, it can be a discrete or a continuous response. With continuous responses, the tangent line at the initial value of the response can be shown on demand; with discrete responses, a zero order hold filter can be applied on demand, Further, a vertical line denoting a specific moment (e.g., the 63%-time) can be displayed.
Specific input description: Taiga provides the minimal and maximal response values to show, the position of the vertical line and a decision as to show a tangent line or to use a zero order hold filter. The input file consists of a row of values, denoting a series of system output values. If the number of values is less than 25, a discrete response is assumed, else a continuous response is assumed.

Processing: If the minimal and maximal response values are equal, suitable values will be calculated from the response.

Output description: A graphical representation of the desired response file. The window is marked by a frame. With discrete responses, the origin \((t = 0)\) should not be located at the extreme left; some space should be kept free. With plain discrete responses, the response values must be denoted by asterisks. The tangent line is dotted in the same colour as the signal; the vertical line and the axes are dotted and in the frame colour. The used minimum and maximum are returned to Taiga.

Display of pole-zero configurations

Function: The display of a pole-zero configuration in a window on the screen. Apart from the actual poles and zeroes, it is possible to place markers on the real or imaginary axis. It is allowed to draw configurations of systems that cannot be realised in practice.

Specific input descriptions: The pole-zero configuration format of TRIP is too much restricted. It leaves no room for placement of markers, as it only specifies a numerator and denominator for the system. This also means that it is impossible to specify plots in which complex poles or zeroes have no complex conjugates. The systems related to such plots are not realisable, but sometimes the item author wants to be able to do this, for example to show only the top half of the S-plane. Further, a format in which the positions of poles and zeroes are explicitly mentioned will be more convenient for the item author. Therefore a special format for pole-zero configuration files is used. Such a file consists of a number of lines. Each line starts with a code \(P\), \(Z\) or \(M\) for Pole, Zero or Marker, respectively. Following the code (without spaces), the multiplicity can be specified. If omitted, a value of 1 is taken. Then the position is given after one or more spaces. The position can be one real number, in which case a position on the real axis is assumed, or two real numbers separated by a comma, in which case a complex number is assumed.

Processing: The scales of real and imaginary axes are calculated according to the available poles and zeroes. Then the scales are equalised.
**Output description:** A pole-zero configuration in the specified window. Poles are denoted by crosses, zeroes are denoted by circles. Multiple poles or zeroes are denoted by indexes giving the multiplicity. Markers are small lines perpendicular to one of the axes, accompanied by the value. Imaginary values are followed by $aj$.

**Display of Nyquist diagrams**

**Function:** The display of a Nyquist plot in a window on the screen. If desired the point (-1,0) is also displayed in the figure.

**Specific input description:** Taiga provides minima and maxima for the real and imaginary axes and a decision as to show the (-1,0) point. The input file has the TRIP format of frequency responses. It consists of three columns, placed serially next to each other: one with moduli, one with arguments, and one with frequencies. Entries with the same offset from the start of the columns form triplets with information about one point in the frequency domain.

**Processing:** the data will have to be converted from pairs of modulus and argument values to pairs of real and imaginary values. If minima and maxima of one of the axis are equal, suitable maxima and minima for both real and imaginary values are calculated, and some space around these is provided. To deal with machine inaccuracy, the difference in scales between real and imaginary axis is limited to 100.

**Output description:** a Nyquist plot of the desired frequency response file. The window is marked by a frame. If the real or imaginary axis is within the viewport, it is displayed as a dotted line in the frame colour. The used minima and maxima are returned to Taiga.

**Display of Bode diagrams**

**Function:** The display of a Bode plot in a window on the screen. Both an exact and an asymptotic plot can be shown. The sizes of the modulus and argument graphs can be varied; one of them can be omitted.

**Specific input description:** Taiga will provide the minima and maxima for the frequency, modulus and argument, and the fraction of the window in use for the argument part. The data file for exact plots uses the TRIP format (see “Display of Nyquist diagrams”). An asymptotic Bode plot of a system is constructed by superpositioning the Bode plots of the individual building blocks: the poles, the zeroes and the gain. So, to be
able to draw an asymptotic Bode plot, the pole-zero configuration is needed, together with the system gain. Therefore, a file format similar to that of pole-zero configurations (see "Display of pole-zero configurations") is used. The markers (code M) are omitted, and a gain can be specified by using the code K. Naturally, K can only have a multiplicity of 1, and to be able to draw an asymptotic Bode plot, the poles and zeroes must all be real-valued.

**Processing:** If the minimum and maximum for some quantity are equal, suitable values are calculated.

**Output description:** The required Bode plot in a window on the screen. Both the modulus plot and the argument plot are marked by a frame. The used minima and maxima are returned to Taiga.

*Display of Mason-type flow diagrams*

**Function:** Display a flow diagram in a window. A flow diagram consists of dots and arrows, both possibly accompanied by a label. The arrows can be straight or curved in a partial circle or arc. The pointer of the arrow is in the middle. Arrow labels are positioned close to the pointer; they can occur at either side of the curve.

**Specific input description:** The input file must allow small variations via parameterised entries. Therefore, a bitmap format will not suffice; a simple vector format is used. First the number of dots is specified, then the positions and labels of the dots can be altered and finally the arrows can be generated. The first is done with the command DOTS (n₁, n₂, n₃, ...). With this, nⱼ specifies the number of dots on line j, counting from the bottom line. The default position of iᵗʰ dot on row j is (i,j). The positions of the dots can be changed freely after the DOTS command, so the multiple numbering is only for convenience, to make a first rough division in dots. It is also possible to enter DOTS (n) for the number of dots, and then to position the dots as desired. This positioning happens via the DOT (nx, label [,x,y]) command. The parameters are the dot number (counted from bottom to top and from left to right, so the first dot on the second row after DOTS(3,5) is 4), the label, in which an underscore means 'no label', and the new position of the dot. If no position change is wanted, the latter two parameters can be omitted.

For the arrows, three commands can be applied: ARROW, LARROW and RARROW. The format of the parameters is (d₁, d₂, label, angle). For all commands, an arrow is drawn from point d₁ to point d₂. If label does not consist of only an underscore, the label is placed close to the arrow, at the right (seen from d₂). If the label should be placed to the left, it must start with a hat (^).
If the command specifies a loop (from and to the same dot) then angle specifies the direction of the loop relative to the horizontal axis. With LARROW and RARROW it specifies the arc in degrees (e.g., 180 means half a circle). With LARROW the arrow bends to the left (seen from $\mathcal{C}_1$), with RARROW it bends to the right.

**Processing:** Unused dots are omitted; the virtual boundaries of the figure are calculated by the positions of the dots after the DOT commands. The figure is then scaled to match the desired boundaries.

**Output description:** The flow diagram as specified in the data file. The colour specifications are used for the colours of dots and arrows.

### 6.4.6 External interface requirements

#### User interfaces

Interfacing with the item author occurs only indirectly. In the item, the author specifies the desire to use a function of one of the toolboxes and provides the necessary data. During run time this specification is used to execute the correct function correctly. For more information about this process see “Software interfaces”.

#### Hardware interfaces

For display, the product relies on Taiga’s graphical and textual display interface. This interface uses a GKS variant which is specified in the Taiga manual.

#### Software interfaces

A description of the Items-D type interface is given in Appendix A. This interface uses data files for communication.

The only way to apply external software in Taiga is to link the object code into the runtime system, and to call a procedure. The communication between the code and the Taiga program takes place via the Taiga variables. This is the only method of direct data transfer between Taiga programs and external code.

Theoretically, there are 99 external procedures available. The procedures below number 48 can be assigned a label in the Taiga programs. As this improves readability of Taiga programs, numbers below 48 are preferred. However, several of these
6.5 Design of the Items-D type Interface for Taiga

have already been used for more or less standard extensions such as selection of a background colour, handling of mouse events, display of a status bar and control of various extension boards.

6.4.7 Performance requirements

The display facilities must be fast enough to avoid annoyance. More concretely, this means that a display function must be finished within a second.

6.4.8 Attributes

Security

Because the system is used for training sessions, security is not a major issue. No special efforts will have to be taken for data encryption, although the possibility for future implementation of security measures must be kept open.

Availability

The system will be made available to students in two ways: they can use it on the local PC net, or they can take it home to use on their own PCs.

6.5 Design of the Items-D type Interface for Taiga

In the description of the design, two types of diagrams are frequently used: dataflow diagrams (DFD) and structure charts (SC). A dataflow diagram shows the data flows in a given system, and the transformations that are applied to the data. Structure charts show the structure of the software system in terms of modules and relations between the different modules of the system.

For reasons of chart clarity, the names of data flow components in structure charts will be shortened to one or two characters; explanations will be given in the text.

6.5.1 High-level structure

I²T can be seen as a library of functions operating on shared data. Figure 6.4 shows a high level dataflow diagram for I²T. The trivial transformation process of the command to one of the described functions has been omitted, and the dataflow names
have been omitted as well; they will be elaborated in the refinement step. The method used for exchange of information, as imposed by Taiga, is made explicit by means of the datastore "Environment variable space".

Data Dictionary

From the interface definition in Appendix A together with the requirements specifications, a data structure for the item data can be derived. The bold entries denote data which can be found in the dataflow diagram; others are refinements.

Item data - Internal representation of the current situation within an item.
   List of Answers
   default Penalty specification
6.5 Design of the Items-D type Interface for Taiga

**Item information** - Item information file as described in Appendix A, section A.1.

**Item result** - Item result file as described in Appendix A, section A.2.

**Data file prototypes** - Data files with specifications for insertion of variables.

**Answer** - Specification of answer alternatives for one subquestion.
  - identification: text
  - type of question: integer
  - maximal score: floating point
  - list of *Feedback information*

**Penalty specification**
  - user score: floating point
  - user answer: text
  - status: (correct, incorrect, too often incorrect)
  - number of mistakes: integer

**Penalty specification** - Description of subtraction percentages at certain events.
  - number of mistakes before too often incorrect (1, 2 or 3)
  - array of subtraction percentages for error 1, 2, 3: floating points
  - list of *Other influencing factors*

**Other influencing factor** - Not error related, score influencing event.
  - key: text
  - subtraction percentage: floating point
  - applied: Boolean

**Feedback information** - Data to calculate the correct answer and to give feedback.
  - lowest allowable answer: text
  - optimal answer: text
  - highest allowable answer: text
  - score: floating point
  - feedback text: text

6.5.2 Function refinements

In the previous section, the high level structure was described. This section handles further refinements for each of the functions, if necessary.
1. Get item data

Item data consists of two parts: static and the dynamic information. Static information is unchangeable during the answering process and is provided by the item information file; dynamic information changes during the answering process and is stored to and retrieved from the item result file. Figure 6.5 shows the data flow. The first step is initialisation of the item data structure. Then the static information (parameters, answer names, maximum scores, penalties for help or errors, etc.) is read. The parameters are transferred directly to the environment variable space, which makes them available in the authoring environment. If the static information contains descriptions of data file prototypes, these are extracted.

If a previous result is available (consisting of dynamic data, such as answers, obtained scores, and used facilities), that result is read into the data structure as well.

2. Create data files

Data files are created by using the data file prototypes, if any, made by the function Get item data. Variable specifications are replaced by current values of the corresponding variables in the environment. See figure 6.6. Variables can be integers, floating points or strings. For floating points the scientific notation of mantissa and exponent is applied. Because it is sometimes necessary to insert carriage returns, and it is not always possible to represent these in a string, a special character is used for this purpose: the "l".
3. Get item status

The function get item status, depicted in figure 6.7, uses the command line (a text variable) to extract the answer identification and the variable names to put the status information in. Using the answer identification, the information is retrieved from the item data; the information components as described in the requirements are assigned to the desired variables. The status can be finished or not finished. The best (correct) answer is defined as the alternative giving the highest score.
4. Check answer

The function Check answer checks the correctness of given answers. First the necessary identifications are extracted from the command line, then the value corresponding to the answer is retrieved from the environment. Then the answer is stored using the identification, the answer is compared to the available alternatives and a score is assigned. When a score is assigned, all score influencing factors are taken into account. According to the Items-D specification, a numerical answer alternative is defined by three values: the smallest allowable value, the actual value and the largest allowable value.

The actual value can be used for feedback purposes. With ordinal or numerical answers, the limits are used to determine whether an answer belongs to an alternative. The limits can be infinite values, denoted by stars: for a minimal value, minus infinity is taken, for the maximal value, plus infinity. With text answers there is no clear notion of greater than and smaller than. Therefore, in such a situation the answer matches if it equals one of the three values.

When processing an answer, the list of alternatives is checked in order of appearance in the item information file; the first alternative matching the answer is taken. This means that alternatives earlier in the list can be used as sieves for those later in the list. By defining a final alternative with infinite limits, all answers not previously caught are handled in the final alternative.

After the alternative has been selected, the update process starts. If a score is assigned that is greater than zero, the status becomes correct, finished. If the score is zero, the number of attempts is incremented. If the number of attempts has become equal to the maximum, the status becomes incorrect, finished, else it becomes incorrect, not finished. If a feedback line occurs with the alternative, it is passed with the answer info as well. The final part is to make the answer information available for the environment; this is done by assigning all components of the answer info to variables for which the names have been given in the command line. The status is transferred in two Boolean variables, one denoting if the answer was correct, and one denoting if it is allowed to ask the question again. With this construct, it is simple to make items that are independent of the number of times a question can be asked. The dataflow diagram is shown in figure 6.8.

5. Register answer

If the item author wants to have control over the answer checking process, it is possible to simply pass the answer and the assigned score to the interface program.
6.5 Design of the Items-D type Interface for Taiga

4

4.4. check answer

item data

answer info

4.5. assign values

IDs for answer info

command line

Environment variable space

4.3. store answer

ID for answer

4.1. get IDs

4.2. get value

answer

Figure 6.8. DFD for “check answer”

After this, the status is automatically set to finished. Using this function to pass an answer implies that other score influencing factors, as described in the following section, are not applied; the question score passed is returned to the management system. Figure 6.9 shows a refinement of the function register answer.

5

5.1. get IDs

item data

ID for answer

command line

Environment variable space

5.2. store answer

5.3. get values

IDs for answer and score

answer and score

Figure 6.9. DFD for “register answer”
6. Register factor

If a score influencing situation occurs in an item, this can be reported by using the function register factor. The command line contains an identification of the sub-question for which the factor applies, and an identification of the factor. This information is used to find the corresponding answer specification in the data, and to store that the factor was used. The information, together with the actual penalty which was read from the item data file, can be used by the answer checking process to obtain the correct score. See figure 6.10.

![Diagram](image)

Figure 6.10. DFD for "register factor"

7. Save result

The function save result saves the dynamic data into the item result file. First the total score is calculated by summing the scores for the subquestions, then the item status is determined; if all subquestions are finished, the item is finished, else if at least one subquestion has been answered, the item is aborted, else the item is not done. This global information is saved into the result file. Then, for each subquestion, the results (answer, status, score, number of mistakes, used influencing factors) are saved. This saving can be done at any time; if a previous result file exists, it is overwritten.
6.6 Design of the Control Toolbox for Taiga

6.6.1 Interfacing with the environment

The Control Toolbox consists of independent functions; there is no high level structure to connect them. Therefore the functions will be described as separate "mini systems". Although the functions are independent, interfacing with the environment takes place in a comparable manner, shown in figure 6.11.

![Diagram of DFD for interfacing with the environment](image)

Figure 6.11. DFD for interfacing with the environment

Data dictionary

*draw information* - Data needed for a drawing function.
  - general draw information
  - specific draw information

*draw result* - Data to be returned to the environment.
  - *computable parameters*
  - status: Boolean

*general draw information* - Data identical to all drawing functions.
  - file name: text
  - *frame*
    - colour 1: colour id
    - colour 2: colour id

*frame* - The section of the screen to draw in.
  - position: 2d floating point
  - size: 2d floating point
2d floating point - A two dimensional value.
   x: floating point
   y: floating point

specific draw information - Specific information for one function.
   computable parameters
   non-computable parameters

computable parameters - Parameters for which defaults can be calculated
   (to be elaborated with each function)

non-computable parameters - Parameters which need to be specified
   (to be elaborated with each function)

The status gives information about the availability of the desired file. If the file
could not be found, the status is false.

6.6.2 Display of time responses

From the function description in section 6.4.5, a number of sub-tasks can be specified. The DFD in figure 6.12 shows the relation between these tasks. The plot limits are extracted from the environment variable space (draw information); if they are equal, the limits are calculated by finding the maximal and minimal values from the plot data file. Further, the number of data points is extracted. Then axes are drawn, if they are within the defined window. Then the plot values are read from file, limited to fit within the window and converted to screen positions. The first two values are saved; they are needed if a tangent line is desired. If the number of values is less than a prespecified number (currently 25), then a discrete graph is shown, else a continuous graph. Drawing a continuous graph is quite straightforward: a polyline can be used. The x-values can be calculated by scaling. If a tangent line is wanted, it can be calculated by extrapolating the first two values. The origin forms the starting point, the intersection with the border forms the end point.

A discrete plot is drawn differently. The x-values are scaled over the available width, but one extra x-position is added. If a zero-order-hold graph is wanted, a polyline is made. For each x-value two points are used, one with the arriving value and one with the departing value. The arriving value is equal to the departing value of the previous point. If a purely discrete plot is wanted, the positions are marked by asterisks (*).
The last action is to draw a border around the figure. A side effect of this is that clipped values, which were drawn on the position of the border, are overwritten by the border.

*Extended data dictionary*

*frame colour* - Colour 1.

*plot colour* - Colour 2.

*file info* - Information about the file to process.
   - file name: text
   - position: long integer

*limits* - Limit values for the axes.
   - maximum time: integer
   - min. response value: floating point
   - max. response value: floating point
Environments and tools for item implementation

**switch** - Boolean.

**plot data** - The response translated to proper coordinates.
   px: array of floating point
   py: array of floating point

**tangent data** - Information needed to calculate a tangent line.
   first response value: floating point
   second response value: floating point

**x position** - Position on x-axis where a vertical line is to be drawn.

**Structure**

The function *draw axes* can be divided into a sub-function to draw horizontal axes and a sub-function to draw vertical axes. This last function can then also be used for drawing the vertical line. A sub-function to draw a tangent line is grouped with the function *draw continuous*. A structure chart is shown in figure 6.13.

![Structure chart for "draw time response"](image)

*Figure 6.13. Structure chart for "draw time response"*

The data flow components are:

- f: file info
- l: limits
- fc: frame colour
- w: frame (window) position and size
- x: x-position
- pc: plot colour
a: arrays with plot data    t: tangent data
b: Boolean value (switch)  st: status

6.6.3 Display of pole-zero configurations

After reading the usual inputs file name, border and colours, the limits are determined by calculating the smallest and largest real and imaginary values from the poles, zeroes and markers in the file. If the file could not be opened correctly, a status flag is set.

Then, for each item in the file, the position is calculated and the corresponding figure (cross for poles, circle for zeroes, horizontal line for marker in the imaginary axis, vertical lines for markers on the real axis) is drawn. A marker is accompanied by a value. Poles and zeroes are drawn in the figure colour, markers are drawn in the border colour. See the DFD in figure 6.14.

\[ \text{Figure 6.14. DFD for "draw pole-zero configuration"} \]
Extended data dictionary

*pz* data - List of *pz*.

*pz* - Pole, zero or marker.
   id: P, Z or M
   position: 2d floating point
   multiplicity: integer

frame colour - Colour 1.

plot colour - Colour 2.

file info - Information about the file to process.
   file name: text

limits - Limit values for the axes.
   min. value: 2d floating point
   max. value: 2d floating point

Structure

Also here the function *draw axes* is divided into the two sub-functions *draw horizontal axis* and *draw vertical axis*. See the structure chart in figure 6.15.

![Structure chart](image)

*Figure 6.15. Structure chart for “draw pole-zero configuration”*
Another refinement concerns draw pzmr. This can be divided into three sub-functions draw pole, draw zero and draw marker, each having the same input as the parent function.

The data flow components are

f - file info  
l - limits  
f - frame colour  
st - status  
p - pzmr data  
w - frame (window)  
pc - plot colour

### 6.6.4 Display of frequency responses

At a high level, the functions to draw Nyquist plots and Bode plots are very similar. They are a different representation of the same data, so there is an overlap in functionality. The file must be read and processed, the limits must be determined (either user specified or calculated from data), and the axes, the plot and the border must be drawn. A data flow diagram for both functions is shown in figure 6.16.

![Diagram](image_url)

**Figure 6.16. DFD for “draw frequency response”**
Extended data dictionary

frame colour - Colour 1.

dot colour - Colour 2.

file info - Information about the file to process.
   file name: text

plot data - The response translated to proper coordinates.
   number of data points: integer
   array of data values

limits - Limit values for the axes.

Structure

The basic structure for the frequency response plot functions is shown in figure 6.17.

![Structure chart for "draw frequency response"](image)

Figure 6.17. Structure chart for “draw frequency response”

The data flow components are:

f: file info  w: frame (window) position and size
l: limits    st: status
fc: frame colour pc: plot colour
a: arrays with plot data
Decomposition for display of Nyquist diagrams

Four observations show specific actions for display of Nyquist plots. First, *read data* returns plot data directly usable for *draw plot*. Since the input file uses frequency, modulus and argument triplets, and Nyquist plots are drawn in the complex plane, the data has to be converted to complex values. Therefore, *read response* is divided into a function to read the data file and a function to convert the data. Second, the function *draw axes* can be decomposed into three parts, two being identical to the sub-functions described in section 6.6.2; the third is a function to display the point (-1,0). Third, the function *draw plot* can be divided into a sub-function to draw the curve and a sub-function to draw an arrow that denotes increasing frequency. Fourth, a Boolean input is used to show if the point (-1,0) must be drawn. Figure 6.18 shows the refinements to the structure chart. The Boolean, denoted by *b*, is also an input value for *get limits*, as the decision whether or not to display the -1 point can influence the limits of the graph.

![Diagram](image)

*Figure 6.18. Structure chart refinement for “draw Nyquist plot”*

Decomposition for display of Bode diagrams

With Bode plots, the situation is a little more complicated. The function must be able to display both asymptotic and normal Bode plots. For the latter, the refinement is similar to that described in the previous section. The input data must be converted to a logarithmic scale. For the asymptotic plots, another type of data file is used. This file delivers poles, zeroes and a gain value. Since the transformation process from poles and zeroes to asymptotic Bode plot is nontrivial, it will be described first. The method people normally use is: (a) split the system into base factors, (b) make asymptotic plots for all base factors, and (c) superimpose these to get the plot for the complete system.
The base factors are formed by the individual poles, zeroes and the gain factor. Second order base factors (pairs of complex poles) are not allowed since it is impossible to form an asymptotic estimate of this base factor. For example, the system

$$H(j\omega) = \frac{2 + j\omega}{1 + 3j\omega + 2(j\omega)^2}$$

(6.1)

can be divided into the base factors $1 + \frac{j\omega}{2}, \frac{1}{1 + j\omega}, \frac{1}{1 + 2j\omega}$ and a gain factor 2.

This first step has already been done; the data file contains the system specification in poles, zeroes and a gain factor. The next step is construction of asymptotic Bode plots for the base factors. For example, the plot for the single pole system

$$H(j\omega) = \frac{1}{1 + j\omega\tau}$$

(6.2)

is sketched in figure 6.19. The plot for a single zero system can be obtained by mirroring the plot of the corresponding single pole system with respect to the $\omega$ axis.

![Figure 6.19. Sketch of the asymptotic Bode plot for $1/(1+j\omega\tau)$](image)

For a factor with a negative $\tau$ (i.e., located in the right half plane in the pole-zero plot), only the argument plot is mirrored. The gain factor only influences the modulus with a constant value.

The function create Bode data uses the gain for initial values of the Bode plots. Possible poles or zeroes in the origin influence the initial slope for the modulus and the starting value for the argument. Then, for each base factor in the system, points of interest are added to the plot. These are the points where the derivative of one of the
plots changes, as shown in figure 6.19. For each point, the slope change after the point is recorded. After all factors have been processed, the plots can be easily constructed because the initial values and the consecutive slopes are all known. The structure chart refinement is shown in figure 6.20. The extra data elements are $b$ (for the indicator normal/asymptotic plot) and $p$ (for the pole-zero data).

![Figure 6.20. Structure chart refinement for "draw Bode plot"

6.6.5 Display of Mason-type flow diagrams

The flow diagram drawing facility is quite different from the other facilities until now. It starts, as usual, with extraction of name and colours. Then the input file is parsed. There are three commands to be handled: dots, dot and arrow. The first creates a list of dots and gives them a default position, as specified in the requirements. With the second command, the position and label of the dots can be modified. The dot with the desired number is located in the list, then the position and label are altered.

The first time that the third command (actually one of three commands, narrow, lar-
row and arrow) is issued, the dot positions are considered final. Then, suitable scaling factors are calculated, enabling arrows to be drawn directly. With an arrow command, the begin- and end-dot are translated to screen positions. Then a curve from the starting point to the end point is created as a polyline. In the middle of the polyline, parallel to the line between the two dots, an arrow is inserted. The resulting polyline is drawn, accompanied by a label, if any. After the last arrow command, the dots are drawn. A dot is represented as a filled circle. By drawing the dots last, the arrows stop at the outer border of the dot. When the arrows and dots are given different colours, placing the dots before the arrows would cause the arrows to over-
write a part of the dot, which would give a cluttered result.
Figure 6.21. DFD for "draw flowdiagram"

Figure 6.21 shows the DFD for the function draw flowdiagram. There is no specific draw information. The complexity of the function do arrow justifies another refinement step. See the DFD depicted in figure 6.22. There are three types of arrows: straight arrows, right bent arrows and left bent arrows. Further, a loop must be drawn if the start and end dots are equal. The curve can therefore have three forms:

Figure 6.22. Refinement for "do arrow"
6.6 Design of the Control Toolbox for Taiga

a circular form, a straight line, and a partial circle or arc. Using an angle and the specification to which direction the arc must bend, an arc can be constructed. The angle parameter is used to specify the part of the circle that is situated between the two dots. With a value of 180, half a circle is drawn, with 90 a quarter, etc. If a loop is to be drawn, the angle parameter specifies the angle that the circle tangent line through the dot makes with the horizontal axis. For example, with a value of 0, the centre of the circle is exactly above the dot.

**Extended data dictionary**

**scaling** - Scaling from maximum coordinates in file to frame size.

*2d floating point*

**dot colour** - Colour 1.

**arrow colour** - Colour 2.

**string** - Line of parameters for the action to perform.

1 line of text

A structure chart is shown in figure 6.23.

---

**Figure 6.23. Structure chart for “draw flowdiagram”**
The abbreviations used for data flows are:

f: file info  
s: string

d: dotlist  
p: frame position

si: frame size  
m: scaling (multiplication factor)

ac: arrow colour  
dc: dot colour

a: arrow information  
st: status

6.7 Design of the General Toolbox for Taiga

6.7.1 Display of external raster images

The function to display external raster images uses an interface with Taiga that is comparable to the interfaces of the Control Toolbox functions. Therefore figure 6.11 applies for this function as well. A file using the Graphic Interchange Format (GIF) consists of a number of components. It starts with a signature, a screen description, and optionally a global colour table. After that, images or extensions may follow. A GIF file can contain more than one image. Extensions can be skipped as they consist of, for example, texts and comments. Figure 6.24 shows a first data flow diagram. An image starts with an image description, and optionally a local colour table. This is mandatory if no global colour table was specified. With the image description and the frame size, a scale factor can be calculated to fit the image in the window specified by the item author. The final part of an image is the actual raster data. The sub-functions to process an image are shown in figure 6.25. To reduce the data size of

![Diagram](image)

*Figure 6.24. DFD for “display GIF image”*
the program, only one colour table is saved. For single-image files, which are the intended files for the function, this will have no consequences. It will only be noticed if there are more images in the file, an early picture overwrites the global colour table with its local colour table, and a later picture uses the global table.

The raster data uses a compression technique known as Lempel-Ziv-Welch (LZW). More information about this technique can be found in the GIF Programming Reference [CompuServe, 1990]. Figure 6.26 shows a refinement for the function read

Figure 6.26. Refinement for “read raster”
Environments and tools for item implementation

*raster*. Decoding the data results in a stream of pixel colours which can be output to the window on the screen using the screen and image descriptions.

**Data dictionary**

*image stream* - Data source for the image.
  - *file name*: string
  - *position*: long integer

*screen description* - Description of the screen to be used.
  - *width*, *height*: words
  - *colour resolution*: byte
  - *global colour table info*
  - array of colour indices (bytes)
  - *background*: colour index (byte)

*colour table info* - Description of used colours.
  - *used*: Boolean
  - *size*: byte
  - *table*: array of colour indices (bytes)

*image description* - Description of current image.
  - *left*, *top*: words
  - *width*, *height*: words
  - *interlace*: Boolean
  - *local colour table info*

*code* - Item of encoded data: word.

*pixel colour* - Colour index: byte.

**Structure**

The structure of the function is shown in figure 6.27.

The abbreviations used for data flows are:

- *f*: file (image stream)
- *s*: screen description
- *ok*: result of signature test
- *w*: frame (window) position, size and scaling
- *i*: image description
- *st*: status

132
6.7 Design of the General Toolbox for Taiga

![Diagram of display GIF image](image)

*Figure 6.27. Structure chart for “display GIF image”*

6.7.2 Display of external texts

The function to display external texts is quite straightforward. First, text is skipped until the required number of page separators have been found. Then, characters are appended to a string until an end-of-line or an escape is encountered, or until the text is about to fall outside the screen. Then the string is displayed on the screen. If an escape character was found, the graphic attributes are set according to the information following the escape character; if an end-of-line character was found or if the string was too long, the next line from the text is taken.

**Structure**

A structure chart is given in figure 6.28.

The abbreviations used for data flows are:

- f: file info
- p: position (text coordinates)
- s: string
- n: page number, starting at 0
- st: status

133
6.8 Example items in Taiga

About 20 different items, each with 5 parallel problems, so making a total of 100 items, have been realised for the course Systems and Control Engineering. Figure 6.29 shows an example question using the facility for drawing time responses. The question tests the student’s ability to make the link between the time constant of a general first-order system (denoted by the tangent line in the beginning) and the starting point of the response of such a system with an initial value of zero. Comparing that point with that of the given system, it is possible to deduce its initial value.

**Given:** From the minimum phase system shown left the impulse response \( x(t) = \delta(t) \) for \( t=0 \) is known for \( t > 0^+ \).

**Question:** What is the initial value at the moment that \( \delta(t) \) occurs, if it is also known that after a unit step \( \lim_{t \to \infty} y(t) = x(t) \)

**Answer:** \( y(0) = 1 \) (integer)
6.8 Example items in Taiga

Given: The measured frequency response of a system:

\[ H(s) = H_1(s) \cdot H_2(s), \text{ which is} \]

the transfer function of \( H_2 \) if

\[ H_1(s) = \frac{10(s + 1)}{s(s + 10)} \]

Answer: \( H_2(s) = \frac{K'}{s} - \frac{K'}{s + a} \cdot \frac{s + a}{s + b} \cdot \frac{K'}{s} \) None of the formulae apply

Enter your answer by moving the highlighted block with the arrow keys and pressing Enter on the desired choice.

Figure 6.30. An item using a Nyquist diagram

Figure 6.30 shows one of the versions of an item using a Nyquist diagram. This item tests the student's ability to reconstruct the building blocks of a transfer function from a Nyquist diagram. For each of the five answer alternatives there is a response file. A parameter containing the desired alternative number is passed to the item; the item displays the corresponding figure and knows which answer is correct.

An example problem using the Mason-type flow diagram drawing facility is shown in figure 6.31. In this problem, the student is asked to find the determinant corresponding to a given flow diagram and to make a choice from the given five alternatives. Variations of this problem are easily prepared. There are five answer

Given: From a system the following flow diagram is known:

\[ \begin{align*}
\Delta &= 1 - g_1 g_2 - g_1 g_3 - g_2 g_4 - g_1 g_2 g_3 g_4 \\
\Delta &= 1 - g_1 g_2 - g_3 g_4 - g_1 g_2 g_3 g_4 \\
\Delta &= 1 - g_1 g_2 - g_2 g_3 - g_2 g_4 - g_1 g_2 g_3 g_4 \\
\Delta &= 1 - g_1 g_4 - g_2 g_4 - g_2 g_3 - g_1 g_2 g_3 g_4 \\
\text{None of the above solutions is satisfactory.}
\end{align*} \]

Figure 6.31. An example item using a flow diagram
Environments and tools for item implementation

possibilities and five diagram files. Each diagram leads to another answer. The teacher must only specify the name of the diagram file to be used, and the corresponding correct answer.

6.9 Porting the toolboxes to other environments

The toolboxes have been built with portability to other environments in mind. In principle only the communication facilities with the implementation environment would have to be changed. The Items-D type Interface was ported to the MS-Windows-based authoring system Inigo [Inigo, 1994] in less than a week, while most of the time was needed to solve Windows-related problems. The kernel code remained practically the same. The main difference was that Inigo communicates by posting messages, as opposed to Taiga using shared data, the environment variable space. Consequently, a modification is also needed for retrieving the parameters. The Inigo program must explicitly report which parameters have to be passed in return, as in the external procedures the Inigo variables are unknown and unreachable.

Porting the Control Toolbox requires more changes. Besides the communication changes, the system of drawing graphs is different in Inigo; it relies on the MS-Windows calls. This means that the routines requiring drawing have to be modified. The strategies and structures remain the same, however. The most practical solution is to rewrite the GKS routines.

There is no need to port the General Toolbox to Inigo, because Inigo already supports importing external figures and texts.

6.10 On-line application of simulation software

In engineering, simulation software is used very often. For example, in electronics the circuit simulation program PSPICE is very well known. It is often desired to use this software, either as a tool for the students, to be used simultaneously with the items, or incorporated in the item, to shield the student from program specific commands distracting him/her from the actual problem. For Electronics, the eventual goal is an MS-Windows based system and a separate a PSPICE based toolbox.

Students starting with control engineering courses tend to have trouble seeing the connection between Laplace domain and time domain. A lot of effort is put into making students see the influence of poles and zeroes on the response. Therefore, an item template was made that not only asks the students a question, but also allows them to do simulation experiments. The basic question is: given a certain step
response, enter the corresponding pole-zero configuration and the system gain. The
teacher can specify the question texts, the feedback, and the system transfer function
(the maximum order is 3, which is the largest order used at the course).

To visualise the influence of poles, zeroes and gain, the student can go into an
experimentation mode. There it is possible to define a pole-zero configuration, and
to have the system calculate the corresponding step response. The student can
choose whether the previous response should be erased or not; if it is not erased, the
differences usually become clearer. The poles and zeroes can be moved by dragging
them to another position in the s-plane. Placing or removing poles and zeroes is
done with a menu. Unrealisable systems are avoided by prohibiting actions leading
to unrealisable systems, such as adding a zero if there are as many poles as zeroes
already. When moving a complex pole, its complex conjugate is moved to the cor-
rect position automatically. An example item is shown in figure 6.32. Students can
change the time scale to zoom in on the initial value, or to zoom out and see the final
value. For such changes, new responses are calculated automatically. Further, the
limits of the y-values can be adjusted. When the student clicks with the mouse in the
response area, the corresponding coordinates (time and y-value) are displayed.

![Pole-zero plot and step response](image)

**Figure 6.32. An item made with the time response template**

Since it is not possible to use external software in Taiga, the item was implemented
in the general programming language Pascal. The user interface was made with a
graphical user interface tool. This tool works as an event handler: it passes messages
denoting occurrence of an event, plus the necessary parameters. It is up to the soft-
ware designer whether or not to act upon those messages. For example, in experi-
mentation mode, the action performed after the message *ok button pressed* is: get
the actual positions of the poles and zeroes, the gain factor, and the final time, calcu-
late the response, and show the result. If a different value for $Y_{max}$ was entered, the action is to display a scaled version of the response. After the slide bar for $K$ has been changed, the action is to update the current value of the gain. There is also an initial event, allowing actions to be performed before the first user action. This is used to process the input files and to calculate the initial response.

The TRansformation and Identification Program TRIP [van den Bosch & Butler, 1990] calculates transformations between different representations of systems. In this template, TRIP is used to calculate a time response from a pole-zero configuration, but it is also possible to create frequency response plots and root loci, to name a few. A version of TRIP not giving graphical output is used as a simulation engine. The platform is MS-DOS; this platform does not allow concurrent presence of two applications. Therefore, a swapper was used to be able to run TRIP. The principle of a swapper was described in section 5.6. The next step is to make TRIP know what its task is without user interference. TRIP is capable of executing a command file at start-up. This feature can be used: when a response has to be displayed, the procedure is to generate a command file and the necessary input file(s), and to let TRIP perform the transformation and store the result into one or more files. After TRIP has finished, the response files are read and the result is displayed. This procedure is used both to show the initial time-response and to show the student initiated time-responses.

Communication with a session manager takes place via the Items-D method of data files and result files. These items are therefore easily applicable in Items-D or in the system described in Chapters 4 and 5.

The item template was demonstrated to students during a course. They judged it to be very clarifying; after the course many students came to play with the program, do some variants and discuss about it. Especially the experimentation mode was appreciated, because the influence of poles and zeroes is visualised very clearly when a next response can be drawn in the same figure as the previous one.

The template can be easily modified into another type of conversion-item, for example, a discrete variant, or an item about the connection between pole-zero configuration and frequency response. Needed changes are in the interface interpretation (i.e., meaning of the parameters and answers), the TRIP command file, and the display facilities.

A drawback of these types of items is that they are made in a general programming language. This means that teachers often cannot make modifications or other variants of the templates themselves. However, they can change characteristics of the system to be displayed, item texts and feedbacks, so a large number of variants can be generated. Variation of parameters is directly implemented as the positions of
poles and zeroes and the gain are passed to the item by the interface file. The teacher will have to take care that the variations do not change the system characteristics too much. For example, a zero on the right half plane gives a non-minimum phase system, which is usually considered more difficult than minimum phase system.

6.11 Conclusion

For item implementation in the pilot courses on control engineering and electronics, the MS-DOS-based authoring system Taiga was used most often. It is quite easy to learn, although several components are somewhat archaic. Since Taiga was lacking vital functionality, extensions have been realised.

The Items-D type interface for Taiga exempts the item implementer from much routine work. Parameters and answer information are read in a single command, the correctness of answers can be checked internally in the interface, and the scores are maintained as well.

The General Toolbox for Taiga provides some generally applicable functions, which were lacking in Taiga. Especially the external graph display routine frees the item implementer of much work, since figures can be reused in other items and they can be realised in any drawing program the implementer feels comfortable with.

The Control Toolbox was developed to place facilities to draw various control theory related graphs at the disposal of item implementers. Other graphs such as block diagrams and other system descriptions can be imported. Many graphs are used in other technical courses as well. For example, in electronics, Bode plots are used frequently. This makes the toolbox is more generally applicable than the name would suggest.

The Items-D type interface appeared to be easily portable to other item implementation environments. An interface library for Turbo Pascal was made successfully as well as a version for the MS-Windows based authoring system Inigo.

A dynamic item template to train recognition of the relation between time responses and pole-zero configurations was implemented in Pascal. The template also allows student experimentation. The time responses are calculated via on-line execution of an external simulation program. The principle is also usable for other types of items in which a connection between two domains has to be recognised.
Environments and tools for item implementation
Experiments and results

Two ‘field tests’ were performed, both within the context of the first-year course in electronics. The purpose of the first test was to investigate the students’ appreciation of the items made by using Taiga and the extensions described in the previous chapter, and to test the performance of the extensions. The choice of items was student controlled: the items could be chosen via a menu. The purpose of the second test was to test the performance of the adaptive mechanism described in Chapter 5, and to see the students’ reactions to the system. First the test set-ups will be described, then the quantitative and qualitative results. The chapter ends with conclusions and suggestions.

7.1 Test set-ups

This section describes item characteristics for the tests, followed by the test situation. The two tests are then handled in more detail and are followed by a discussion of the questionnaire used with the second test.

7.1.1 Item characteristics

Content

Both tests used the same types of items, concerning four of the basic subjects in the introductory course on electronics. Figure 7.1 shows these subjects and their relations. The arrows denote dependency on the subject pointed to, the numbers show the preferred curricular order. This relational figure was given by the course lecturer.
Experiments and results

Figure 7.1. Relations between the subjects in the test

Structure

The items all use a structure which is aimed at the encouragement of active participation. Therefore the solution is not available until the student answers correctly (to allow the student to see if the correct line of reasoning was followed) or gives too many incorrect answers.

Since all items in the examination on the course Electronics have a multiple choice form (usually six choices), this form was maintained in the item questions. However, it can easily be replaced by another type such as a numerical variant.

After an incorrect answer, a cue is given, hopefully leading the student in the right direction without immediately revealing the correct answer. The cue to be given with an incorrect answer usually depends on the answer given. The first error could be a small mistake, but after more errors it is more likely that a student is guessing. As the probability of guessing correctly increases with multiple choice items, only one extra attempt is allowed. However, if numerical items were used, more attempts could be allowed. The general structure of the items is given in figure 7.2.

Layout

The problem presentation consists of text (possibly including formulae) and usually a drawing of a circuit or a diagram. As it is important that the student focuses on the problem, the layout must be clear. With a standard layout for items, students will know immediately where to expect what.

When giving the solution, it is considered important that the question is still visible, to prevent putting a large burden on the student's short-term memory.
Further, the layout must take into account that an option in a multiple choice item can take up to a complete line (about 75 characters). A final consideration is that some items do not use images, but use more text. From these considerations a standard layout was designed as shown in figure 7.3.

### 7.1.2 Test situation

In both tests, the course had already been finished; the sessions were held after the course examination. The participating students were randomly selected from the complete group, thus including students who had passed the examination, students
who had failed, and students who had not yet taken the examination. The sessions were included as a sub-task in the first-year electronics practical; this implies that the students were obligated to do it, because a practical can only be passed if all sub-tasks have been finished. However, the students were told in advance that (serious) participation implied passing that particular sub-task; no minimal scores had to be reached, for example.

7.1.3 Functionality test of the implementation environment

A non-adaptive session with a small number of students was done at an early stage in the implementation phase. Its goals were

- to see if the communication between the items and the management system was functioning properly in practice,

- to see if the environments and their extensions behaved as expected, and

- to involve students at an early stage in order to be able to take their reactions, criticism and suggestions into account.

Four students, two lecturers, a student assistant and the practical supervisor participated. As session manager, the package Items-D was chosen. Two sessions with 10 items each were offered; the user had the opportunity to choose one of the two, or both. One was more difficult than the other, offering a bit of student control. Further, the students could choose the sequence of the items in a session as items in each ses-
sion could be chosen via a menu. After the session, the test was concluded with an evaluating discussion. Because of the small number of participants, a questionnaire would not give significant results.

7.1.4 Functionality test of the adaptive system

In the second experiment, an item bank of 48 items was available, divided over the 4 subjects shown in figure 7.1. The adaptive session manager, as described in Chapter 5, was used. The item bank included most of the items from the first experiment; a number of items were modified when the evaluation of the first experiment gave reason for doing so. For example, in some items, the solutions or the figures were found to be unclear, sometimes the colours or underlinings were not used consistently. The goals of the experiment were

- to evaluate the student’s appreciation of the system, with emphasis on the adaptive session manager on the one hand and the item layout and clearness on the other.

- to see if the system adapts in a reasonable way to the student’s level.

Three groups of about 10 students participated in this test. The selection can be seen as random because students of those practical groups who happened to do their practicals during a certain period were selected.

One afternoon was reserved for each group. First, the lecturer explained the purpose and structure of the session, then the students did a training session with the system for approximately 90 minutes. After this the students were asked to fill in a questionnaire, which is described in the next section. The test session was concluded with an evaluating discussion.

7.1.5 The questionnaire with the second test

To be able to compare the questionnaire results with the session results, a unique identification was used to identify the students. It was suggested to take the student numbers as this is a unique identification, but students were free to take another number, should they consider the use of their student number an infringement of privacy. The only requirements on the students were that they used the same number in the session and in the questionnaire and that they could not use a number which had been used already by someone else, as the identification must be unique.

The questionnaire consists of six parts.
The adaptive session manager uses an estimate of the student's ability to adapt the level of the items. To test the accuracy of this estimate, a reference is needed. In the first section (questions 1 to 6), some information is gathered in order to form an impression about the ability and other characteristics. Because the experiment was done at the end of the curriculum year, most students had already done their examination; this mark is a good starting point for comparison. Further, the students had done the prerequisite course "Electrical Circuits". To gain more insight into the student characteristics, also the amount of time spent on home study, the number of lectures followed and the number of instruction sessions followed are asked. The last two factors can give relevant information because attending lectures and instruction sessions is not obligatory at Dutch universities.

The second section (questions 7 to 11) concerns the exercise in general. It is meant to examine whether the session manager actions were found clear, logical and acceptably fast.

The third section (questions 12 to 17) focuses on the items themselves, in particular the layout, the clearness of hints and solutions and the speed of image composition on the screen. This is used for the evaluation of the item implementation environments and the items, and to see if the student's judgments about the complete system are influenced by the items.

Then with the fourth (questions 18 to 20) and fifth section (questions 21 to 23) an idea of the student's appreciation of the adaptive system is formed. The fourth section concentrates on the succession of items, i.e., whether or not the subject and level chosen by the system seemed logical, if the principle appealed, and if some more influence is desired. In the fifth section some motivation of the judgement about the principle is asked.

The sixth section (questions 24 and 25) asks for a general opinion about the method of training. An extra question concerns the student's opinion about using a comparable system for adaptive examination.

Finally, the student is asked to make remarks and suggestions for improvement.

The original Dutch questionnaire and an English translation can be found in Appendix B.
7.2 Statistical analysis of adaptive system test

7.2.1 Introduction

Using the statistical package SPSS, an analysis of the questionnaire data and the session results was performed. As the number of participating students was not very high, the results only give an indication. Although strictly it is not entirely correct with non-numerical values, the means for all students of each answer to the questionnaire give a global impression of the students' opinions. This is elaborated in section 7.2.2.

One of the objectives was to investigate if there is a relation between the student's ability and its estimate by the system. As a measure for the ability, the first section of the questionnaire data was used. Further we wanted to reveal other possible relations in the questionnaire. The analysis results are described in section 7.2.3. The session result is analysed in more detail in section 7.2.4. Finally, item occurrences in the sessions are handled in section 7.2.5.

7.2.2 Global indication of questionnaire result

In the Netherlands, examination marks can take values from 1.0 (extremely bad) to 10.0 (excellent); usually 5.5 is the minimum to pass the exam.

If the answers for the agreed / disagreed bars (questions 7 to 25) are translated to a value between 1 and 5, in which 1 corresponds to “disagree”, 3 corresponds to “neutral” and 5 corresponds to “agree”, the means can be calculated. The results for the complete questionnaire (except question 1 which asked about pre-university education; this cannot be transformed to a numerical value) are shown in table 7.1.

<table>
<thead>
<tr>
<th>Question</th>
<th>Description</th>
<th>Cases</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>mark for Electronics</td>
<td>27</td>
<td>6.74</td>
</tr>
<tr>
<td>3</td>
<td>mark for Electrical Circuits</td>
<td>28</td>
<td>7.39</td>
</tr>
<tr>
<td>4</td>
<td>time used for home study</td>
<td>28</td>
<td>21.2</td>
</tr>
<tr>
<td>5</td>
<td>proportion of lectures followed</td>
<td>28</td>
<td>0.77</td>
</tr>
<tr>
<td>6</td>
<td>number of instructions followed</td>
<td>28</td>
<td>5.07</td>
</tr>
<tr>
<td>7</td>
<td>fit with course matter</td>
<td>28</td>
<td>4.37</td>
</tr>
<tr>
<td>8</td>
<td>clear what expected</td>
<td>28</td>
<td>4.60</td>
</tr>
</tbody>
</table>
Table 7.1. Means of the answers in the questionnaire (continued)

<table>
<thead>
<tr>
<th>Question</th>
<th>Description</th>
<th>Cases</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>program actions logical</td>
<td>28</td>
<td>4.35</td>
</tr>
<tr>
<td>10</td>
<td>execution fast enough</td>
<td>27</td>
<td>4.71</td>
</tr>
<tr>
<td>11</td>
<td>survey gives good impression</td>
<td>28</td>
<td>3.94</td>
</tr>
<tr>
<td>12</td>
<td>easy to enter answer</td>
<td>28</td>
<td>4.43</td>
</tr>
<tr>
<td>13</td>
<td>item layout is clear</td>
<td>28</td>
<td>4.26</td>
</tr>
<tr>
<td>14</td>
<td>hints and remarks are clear</td>
<td>26</td>
<td>3.90</td>
</tr>
<tr>
<td>15</td>
<td>solutions are clear</td>
<td>27</td>
<td>3.84</td>
</tr>
<tr>
<td>16</td>
<td>extensive elaboration preferred</td>
<td>27</td>
<td>3.70</td>
</tr>
<tr>
<td>17</td>
<td>display construction fast enough</td>
<td>28</td>
<td>4.39</td>
</tr>
<tr>
<td>18</td>
<td>subject &amp; level choices logical</td>
<td>28</td>
<td>4.04</td>
</tr>
<tr>
<td>19</td>
<td>want to be able to indicate subjects</td>
<td>28</td>
<td>4.34</td>
</tr>
<tr>
<td>20</td>
<td>adaptive principle appeals</td>
<td>28</td>
<td>4.11</td>
</tr>
<tr>
<td>21</td>
<td>tuning level is handy</td>
<td>28</td>
<td>4.34</td>
</tr>
<tr>
<td>22</td>
<td>want to say what problem next</td>
<td>28</td>
<td>3.22</td>
</tr>
<tr>
<td>23</td>
<td>different items with different sessions</td>
<td>28</td>
<td>4.39</td>
</tr>
<tr>
<td>24</td>
<td>general judgement</td>
<td>28</td>
<td>4.32</td>
</tr>
<tr>
<td>25</td>
<td>adaptive examination</td>
<td>28</td>
<td>3.41</td>
</tr>
</tbody>
</table>

Although the number of students is not very large, it is safe to say that in general the system was appreciated by the students. The reason for this statement is that the values in questions 20 and 24 are rather high.

The questions not in the first section, with a mean value of less than 4 are considered more carefully, as these are the questions in which the agreement is lower. The following can be noted: the survey of results (between two items) is probably not clear enough (11) and the hints, remarks and solutions could be improved (14 & 15). It is also obvious that the students are not very positive about an adaptive examination (25). Further, many students did not favour more extensive elaborations (16) and are neutral about the ability to say what problem to solve next (22). Many of these results correspond with the impressions obtained during the evaluative discussions. At first glance, the low average of question 22 seems odd, because during the discussions it seemed that people wanted more influence. However, there was also a group of students who liked the system to decide about a next item. Further, most of the students who wanted more influence only wanted to be able to adjust the subject and the starting level of difficulty, and did not desire to say exactly what item to do. This is discussed in more detail in the following sections.
7.2.3 Questionnaire analysis

Correlation analysis

To investigate correlations between student characteristics and the answers to the questionnaire, a correlation analysis was performed between the first section of the questionnaire on one side, and all the questions on the other side. Further, a correlation analysis between questions 20, 24 and all others was performed. It is not unusual to consider the questions interval variables; therefore the Pearson correlation coefficient is used. The coefficients with a 1-tailed significance P (which can be interpreted as the probability that the correspondence is coincidental) of less than 0.01 are shown in table 7.2. The correlation matrix is included in Appendix B.

<table>
<thead>
<tr>
<th>Questions</th>
<th>Correlation</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-3</td>
<td>0.54</td>
<td>0.002</td>
</tr>
<tr>
<td>4-5</td>
<td>0.59</td>
<td>0.000</td>
</tr>
<tr>
<td>4-21</td>
<td>0.44</td>
<td>0.010</td>
</tr>
<tr>
<td>5-6</td>
<td>0.81</td>
<td>0.000</td>
</tr>
<tr>
<td>5-20</td>
<td>0.52</td>
<td>0.002</td>
</tr>
<tr>
<td>5-24</td>
<td>0.68</td>
<td>0.000</td>
</tr>
<tr>
<td>6-24</td>
<td>0.56</td>
<td>0.001</td>
</tr>
<tr>
<td>20-21</td>
<td>0.52</td>
<td>0.003</td>
</tr>
<tr>
<td>20-24</td>
<td>0.49</td>
<td>0.004</td>
</tr>
</tbody>
</table>

Stressing again that the number of students is too low to speak about proofs, this table shows trends that can be verbalized as follows for some of the relevant factors:

- People who do a lot of home study like the idea that the system tunes the difficulty to performance (4 - 21).
- People who follow the lectures also tend to follow the instructions (5 - 6), and do much home study (4 - 5).
- People who followed many lectures tend to like the principle of the system better and judged the way of training more positively than people who followed less lectures (5 - 20 and 5 - 24).
- People who followed more instruction sessions seem to like the system better than those who followed fewer instruction sessions (6 - 24).
Experiments and results

- The judgement about the system in general is loosely correlated to the principle (20 - 24).

These results are not illogical. Very good students usually do not need a lot of practice and, if they need it, they can decide for themselves what is needed, so they do not benefit from an adaptive training system. The system is more aimed at the moderate students who need much training to master the subject. This group is mainly positive about the system, as can be seen from the correlation between questions 4 and 21.

The significant correlation between questions 20 and 21 indicates that people who like the principle think it is handy that the system adjusts the difficulty level. This is a consistent result. Because there is no significant correlation between 20 and 22 there is no ground to support the idea that students who did not like the principle were motivated by their inability to choose the next item.

T-test analysis

The t-test is a method to investigate if there are differences in means of attributes in two samples. The population is split into three roughly equally large parts, based on two threshold values for a criterion attribute. The group with values between the two thresholds is omitted; the two remaining groups are compared with respect to other attributes. The P-value gives the probability that the populations have equal means for this attribute. If the P-values are small, there is a large probability that the samples have different means for the chosen attribute, and therefore it is likely that there is a dependency between that attribute and the attribute used for separation. T-tests were done with separations based on questions of the first section (2, 4, 5, 6). The means of the answers to all other questions were tested. The results giving a P-value smaller than about 0.1 are given in table 7.3.

<table>
<thead>
<tr>
<th>division based on</th>
<th>applied to</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>0.014</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>0.066</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>0.009</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>0.016</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>0.052</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>0.051</td>
</tr>
</tbody>
</table>
Table 7.3. Significant results of t-tests (continued)

<table>
<thead>
<tr>
<th>division based on</th>
<th>applied to</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>0.084</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>0.001</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>0.103</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>0.099</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>0.077</td>
</tr>
</tbody>
</table>

The procedure is illustrated using the last line in the table. The population was divided into three groups based on their answers to question 6, the number of instruction sessions followed. There is a probability of 7.7% that the differences in opinion about the system (question 24) between the people following little instruction and people following much instruction is due to coincidence.

New results are:

People who follow many lectures appreciate the difficulty level tuning more than people who follow few lectures (5 - 21).

On average, people who follow many instruction sessions perform better in the examination (6 - 2), have less difficulty with item layout (6 - 13) and appreciate the tuning of difficulty level more (6 - 21) than people who do not follow many instruction sessions.

The occurrence of differences in answers to question 21 with separation on questions 4, 5 and 6 show that the difficulty level tuning is appreciated more by students who spend much time studying.

The relation between the amount of instruction followed (6) and the examination mark (2) indicates that the training system can be very useful. Further, it is remarkable that no significant differences in opinions and learning characteristics can be pointed out if the population split is based on the examination marks. However, this could be caused by the small samples.

Some final t-tests were done with separation based on questions 20, 21 and 24, concerning the adaptive principle, the appreciation of the level tuning and the general judgement. The relevant results are summarised in table 7.4, in which relations that were already shown in previous analyses are omitted.

When looking at the general judgement (24), there appears a difference in opinion about the clarity of the survey between two items (11). This could mean that clarity of the survey is apparently important for the final judgement. Students who liked the adaptive principle (20), scored lower on the desire to indicate subjects (19) and the


Table 7.4. Results of t-test based on judgements about the system

<table>
<thead>
<tr>
<th>division based on</th>
<th>applied to</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>19</td>
<td>0.041</td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>0.096</td>
</tr>
<tr>
<td>21</td>
<td>7</td>
<td>0.021</td>
</tr>
<tr>
<td>24</td>
<td>11</td>
<td>0.018</td>
</tr>
</tbody>
</table>

next item (22). Here the relation appears which could not be shown in the correlation analysis. The relation between the opinion about fit with the course matter (7) and handiness of level tuning (21) is a somewhat strange result. It could be explained by assuming students need insight to be able to answer question 7, and people with less insight probably do not appreciate level tuning because either they cannot follow what the system is doing, or they remain at lower levels.

Another result is that no correlation nor other evidence could be found that the student judgement about the system is influenced by the item-specific properties (questions 12 to 17). This could mean that the students' judgement is more based on the management system than on the items, as there are differences in opinion about the items in both the positive and the negative group.

7.2.4 Analysis of session results

Normal methods of determining the reliability are not applicable. These methods assume that every student has had the same items, which is not the case. Both the number of administered items and the difficulty of the items differ, not to mention that there are several subjects. Further, the methods require that the proficiency level of the student does not change during the test. This is not true in a training session. Therefore, other methods of estimating performance of the system have been applied.

Correlation between examination marks and session results

A first test to see if the system has performed reasonably is a correlation analysis between the examination marks and the shown levels as obtained by the system at the end of the sessions. As mentioned before, the method stores two estimates in the student model. When an item has been answered, the level of the item becomes the third value. If the item was answered correctly, the two highest values are stored, else the two lowest values. Estimate 1 is optimistic (the higher level of the two), estimate 2 is pessimistic (the lower level). The final estimate is formed by averaging the two.
7.2 Statistical analysis of adaptive system test

From the session data, it appeared that the estimates are biased because identical items were asked more than once. This is caused by the limited size of the item bank, and by the fact that the items did not yet have varying parameters; if an item was asked another time, it was exactly the same.

The results are shown in table 7.5, in which the stars denote significance (i.e., a P-value of less than 0.01). This table shows that the subject Components correlates

<table>
<thead>
<tr>
<th>Subject level estimate</th>
<th>Electronics</th>
<th>Electrical Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components, estimate 1</td>
<td>0.50*</td>
<td>0.29</td>
</tr>
<tr>
<td>Components, estimate 2</td>
<td>0.51*</td>
<td>0.32</td>
</tr>
<tr>
<td>Diode Circuits, estimate 1</td>
<td>0.14</td>
<td>0.067</td>
</tr>
<tr>
<td>Diode Circuits, estimate 2</td>
<td>0.16</td>
<td>0.059</td>
</tr>
<tr>
<td>CE Model, estimate 1</td>
<td>0.27</td>
<td>0.066</td>
</tr>
<tr>
<td>CE Model, estimate 2</td>
<td>0.27</td>
<td>0.055</td>
</tr>
<tr>
<td>Amplifier Circuits, estimate 1</td>
<td>0.22</td>
<td>0.46*</td>
</tr>
<tr>
<td>Amplifier Circuits, estimate 2</td>
<td>0.17</td>
<td>0.40</td>
</tr>
<tr>
<td>Mean</td>
<td>0.32</td>
<td>0.32</td>
</tr>
</tbody>
</table>

more or less with the examination mark on electronics. The subject Amplifier Circuits is more related to the mark on electrical circuits. Taking into account that most students did not do the second round completely, it seems logical that the Components estimates show the best correlation; many students completed only this subject in the second round. The low correlation in the subject Diode Circuits can be explained by a mistake in two items, which caused the correct answer to be judged as incorrect. Unfortunately, these items had both a fairly low level of difficulty and were given to almost everybody. The influence of this was severe because after an actually correct answer the estimate decreased. To make things worse, because of the limited number of items, the item was likely to be given twice to the same student. Another influence was that because students stayed at the subject Diode Circuits longer than necessary, less time was available for the other subjects.

If the second round was not completed, the estimate returned by the system can be seen as a minimum for the real level. Then, a comparison with a limited score is more appropriate. The state "probably mastery" corresponds to a mark greater than 5; the state "mastery" for greater than 7. Therefore, analyses were done with an upper limit to the examination marks of 6 and 7. It was expected that the results would be especially noticeable with the subject Amplifier Circuits. This appeared to
Experiments and results

be true; it is most clear in the correlation with the examination marks on Electrical Circuit (this is explainable: calculations on amplifier circuits require skills in circuit analysis). Table 7.6 shows the result. In this table, single stars denote P-values of less than 0.01, double stars denote P-values of less than 0.001.

<table>
<thead>
<tr>
<th>maximum filter</th>
<th>amplifier circuits, est. 1</th>
<th>mean of all estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.73**</td>
<td>0.48*</td>
</tr>
<tr>
<td>7</td>
<td>0.67**</td>
<td>0.50*</td>
</tr>
</tbody>
</table>

T-test analysis

As the correlation analysis did not give very significant results, several t-tests have been performed to investigate whether there is a significant difference in the means of system estimates if the population split is based on the examination mark on Electronics. Further, the numbers of answered items on each subject were compared. The t-test was not done for the mark on Electrical Circuits, because it appeared not to be possible to split the group into roughly equally large samples.

<table>
<thead>
<tr>
<th>Applied to</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components, estimate 1</td>
<td>0.007</td>
</tr>
<tr>
<td>CE Model, estimate 1</td>
<td>0.105</td>
</tr>
<tr>
<td>Amplifier Circuits, estimate 1</td>
<td>0.072</td>
</tr>
<tr>
<td>mean estimate over all subjects</td>
<td>0.019</td>
</tr>
<tr>
<td>no. of items, diode circuits</td>
<td>0.044</td>
</tr>
</tbody>
</table>

A remarkable connection is that with the number of items for the subject Diode Circuits, showing that students with high grades have done more items than students with lower grades. One could assume that this is logical because the better students have done two complete rounds. However, weaker students usually make more mistakes, so that it will take more items before the first stage has been reached. A probable cause of the difference is that the lower achievers were caught by the mistake in one of the items, making the estimate drop below minimum, thus causing the program to stop asking about the subject. Better students did not get the item or recognised the mistake, but perhaps lesser students were confused. This hypothesis stresses the importance of correct items.
In summary, it appears that the system is quite capable of discriminating between good and less good students, although there is some noise in the form of mistakes in questions, questions being asked more than once, and level of item difficulty not being known exactly. It must be remembered that the tests were ended after a certain amount of time; in a normal situation the students can continue for as long as they desire (unless the system concludes on either too little knowledge or mastery). In this case, for many students the process was interrupted at a random moment.

### 7.2.5 Analysis of the items asked

Figure 7.4 shows the frequency of occurrence of all 48 items in the item bank in the second test. Per subject the items are ordered on an increasing level of difficulty.

The first and second subjects show one item that has been asked far more than the others. The cause of this is the initial estimate and the small item bank: there is only

![Graphs showing item frequency](image)

*a. Components  b. Diode Circuits  c. CE Model  d. Amplifier Circuits*

*Figure 7.4. Occurrence frequencies of the items in the second test*
Experiments and results

one candidate to be asked initially. This means that all students get that item. With the third subject, there are two candidates (34, 35); this can be seen immediately. It is logical that the difficult items are asked less frequently than the easier items, because the difficulty level increases gradually.

Item 17 judged the correct answer to be incorrect. This explains the high number of incorrect answers and the absence of correct answers in the second attempt. When confronted with a feedback saying the answer was incorrect, people thought they had made a mistake with entering the correct answer and tried the same answer again.

Figure 7.5 shows the numbers of times the items were administered, if only the first occurrence of an item to a student is counted. Item 37 is probably less difficult than expected, as almost everybody answered it correctly.

![Figure 7.5. Occurrence frequencies of items with doubles omitted](image)

Table 7.8 shows the average number of items done per student, and the proportions of correct responses, correct responses after a mistake, and incorrect responses. Multiple occurrences are omitted. The percentage-correct is reasonable; in an optimally discriminating adaptive situation, the number of answers correct in the first attempt would be 50%. Now the adaptation is influenced in a manner that good students do not get hard questions immediately, it seems logical that the proportion of correct answers is higher. The lower percentage in the fourth subject is probably caused by the low number of available items, often causing harder items to be chosen than desired.
Table 7.8. Average session with multiple occurrences of items omitted

<table>
<thead>
<tr>
<th>Subject</th>
<th>No. of items</th>
<th>correct</th>
<th>correct/2</th>
<th>incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components</td>
<td>5.3</td>
<td>68.0%</td>
<td>14.4%</td>
<td>17.6%</td>
</tr>
<tr>
<td>Diode Circuits</td>
<td>4.8</td>
<td>54.0%</td>
<td>22.3%</td>
<td>23.7%</td>
</tr>
<tr>
<td>CE Model</td>
<td>4.4</td>
<td>68.5%</td>
<td>15.7%</td>
<td>15.7%</td>
</tr>
<tr>
<td>Amplifier Circuits</td>
<td>3.8</td>
<td>47.3%</td>
<td>22.7%</td>
<td>30.0%</td>
</tr>
<tr>
<td>total</td>
<td>18.5</td>
<td>60.3%</td>
<td>18.1%</td>
<td>21.6%</td>
</tr>
</tbody>
</table>

Due to the small samples, it is not possible to come to sound conclusions about the estimated item difficulty levels, but in general the results do not indicate that the difficulty levels are much different from the estimated levels.

When taking the total scores without omitting the items, the average number of items is 20.2. The proportions correctness of the second occurrence of items are interesting figures. The percentages are 61.2%, 14.3% and 24.5% for correct, correct the second time and incorrect. The multiple occurrence of items does not seem to have a large impact on the raw scores. This can partly be explained by the fact that the system does not choose an item for a second time unless there is no other option. Therefore it does not happen very often that the same item is asked twice consecutively; if another item is asked in between, the short-term memory has probably forgotten the answer already (although some students made notes when answering questions; the notes could be re-used as well!). However, it should be taken into account that better students do not get the same item twice. If a student keeps on answering correctly, the estimate will keep on increasing, so an item that was best on a previous occasion will not be best on the next. Therefore, the figure actually shows an increase in the percentage correct, if measured on the population sample that had the most chance of getting them. This can have three reasons: either the students remembered the answer or they have written it down, or they have learned how to handle the problem. It is probably a mixture of all three.

7.3 Qualitative analysis of the test results

7.3.1 Technical performance

The system was installed on a network server; the clients were all PCs with an 80486 processor and at least 4 MB of memory. It was a lot more troublesome to install the Items-D session manager than to install the adaptive session manager;
Experiments and results

Items-D demands more specific settings. Installing Items-D on the network server took about 2 days; installing the adaptive session manager took half an hour. The performance was acceptable; items were started within a second after calling, and the screen image composition time hardly ever exceeded a second. The only function that is not very fast is the external graph displayer. However, for one-bit colour images the performance is acceptable. From the 7 participants of the first test, 6 were satisfied with the speed in general, one thought it to be a little slow.

The communication between session manager and item environment Taiga works without problems. That was to be expected as the interface is operational in other departments as well, for example in the Department of Civil Engineering, with a computer-based test system on constructional sciences [Lee, 1992].

7.3.2 The evaluation discussions: student judgements

Item layout

Concerning the opinions about item layout, three groups can be distinguished. The first and largest group (about 60%) found the items clear (except for some minor flaws) and the layout well organised. The second group (about 30%) found the items to be not very clear, this was caused especially by the font of the items (or the absence of a formula editor), which made the formulae in particular somewhat hard to read. Figure 7.6 shows an example. The third group did not give a clear opinion.

\[ r_y = r_f/q_n = 25 \text{ kR}, \quad r_o = u_o / I_c = 300 \text{ kR}. \]

\[ \frac{I_p}{U_s} = \frac{r_y}{r_y + r_o} = \frac{25}{25 + 300} = 3 \times 10^{-3} \text{ A/V} \]

*Figure 7.6. Example of a formula in an item*

Many students of the second group favoured the suggestion of porting the system to an MS-Windows-based environment, because of the many extra possibilities this would give, especially in connection with display of formulae and graphics. Also, with the multitasking facilities, it would be possible to use a notepad and a list of formulae, while now it was necessary to use the book and a piece of paper. But several students in the first group strongly rejected the suggestion of using a system based on MS-Windows because of probable size increase and speed decrease of the software. Further, the system demands when using MS-Windows would be higher.
7.3 Qualitative analysis of the test results

Item structure

Considering the structure of the items, the participants generally appreciated the fact that solutions are not available directly; this is considered stimulating. Some said they were able to concentrate longer than with a book or with the usually used collection of elaborated exams. The students also found it stimulating that the system gives them estimates (seeming opinions) of their knowledge level on a subject. Some students said they were not satisfied until they had "persuaded" the system to move the status from perhaps learned to probably learned or even strong, whereas they said that with classical exercise sessions or at home they cannot bring themselves to practice. Some students found it a drawback that it is impossible to ask for a hint without "losing" one answer opportunity. Further, the items were found somewhat too static. This can also be explained by the lack of parameter changes in the items and the modest set of problems.

Item content

The reactions to the content were for the most part positive. In general, if there was criticism it focused on details. In some cases, the hints were considered too explicit; they did not leave any option other than the correct one. In other cases the solution was found too compact. For example, in one of the items just the formula to be used was given, but it was not clear why specifically that formula had to be applied. These situations were mainly caused by the size restrictions of the feedback window, due to Taiga's limited display area of 80 characters and 25 lines, in combination with the design decision that the question must still be visible if the solution is given.

Management system

In the evaluation of the first test it was asked if an adaptive version of the item selection would be appreciated. Although all the students answered affirmatively, it is a bit premature to conclude that therefore the adaptive system would be better, as the students probably did not exactly know what to expect with an adaptive system. However, it was encouraging us to continue realisation of an adaptive manager.

A better criterion is the judgement of the second group, as they can imagine what a menu-driven non-adaptive system would look like. In the evaluation discussion of the second test, two groups could very clearly be distinguished. One group appreciated the system control, because it enabled them to focus on the problems to solve. They found it very handy for training purposes. The other group wanted to have
more control over the selection of the problem to be solved next, or at least over the strategy the system uses to decide what item to select next. Not being able to choose a subject was particularly seen as a limitation.

A general opinion was that the training becomes tedious after about an hour. Apart from the usual loss of concentration after a period of time — research has shown that in lectures student concentration decreases after 20 minutes — the main reason for this appeared to be the small size of the item bank. Students found the items to be too similar, especially after a few mistakes. It could easily happen that a student got the same item twice. Since the lecturer had not supported variation of parameters yet, this resulted in too little variation.

Both in the open part of the questionnaire and in the discussion, suggestions for improvement of the system were given. These can be summarised as two main points:

- Allow the students to influence the item selection strategy. It was seen as a drawback that it is not possible to enter a strategy, or to overrule the item being chosen. It should be possible to restrict the subjects to train and the maximum level, or to state the start subject of the training. Another suggestion was to support “trial exams”, i.e. give a set of problems which could form an exam.

- Give study advice after the test. The students would appreciate an evaluation with an overview of their weaknesses with a view to what parts of the course need extra attention.

**Summary**

The adaptive system is usable and valuable, and especially well suited for study purposes. However, it needs to be smoothed before it can really be applied on a large scale.

**7.3.3 Lecturer and observer experiences**

**Implementation of items**

During the preparation of the tests, the vast majority of the items had been realised by the lecturer of the Electronics Course. His opinion is that it takes some time to acquire skill in authoring with Taiga but, after that, realising items is quite easy. There are some drawbacks, however. It is not handy that the texts (item text, hints, solution) are scattered all over the item, and the Taiga text editor is rather primitive.
7.3 Qualitative analysis of the test results

Probably the first drawback will be diminished if the text display utility of the General Toolbox is used instead of the normal text frames; then texts can be grouped in one single text file.

**Item selection by the adaptive system**

Confronting the lecturer with the test results (consisting of session overviews with items asked, scores and final estimates) his conclusion was that the system performs "not badly at all". The good students traversed the program quite quickly. For the majority of students, a rough interpretation of the sessions showed that the system estimates were in line with the teacher’s judgement.

He stressed that the levels of difficulty given to items were only based on experience which makes them a bit arbitrary or at least subjective.

Sometimes the system stays too long on one subject. This should be prevented, even if the system is not capable of giving a just judgement. After a number of more or less failed attempts at answering a question, it should be clear to the student that there is a lot of studying left to do. Instead of a new item, explicit study advice would be a better option. The bottom left picture in figure 7.7 (page 163) shows a situation in which the system should have stopped. The problem there is the absence of items with a suitable level of difficulty.

**Information supply**

The judgement about the student’s performance should be more explicit. In that judgement, the number of items asked and the time used are also factors which can be taken into account. For example, a large number of used items could indicate that the student has been guessing. Further, especially when a student wants to test his examination capabilities, speed of answering is a non-negligible factor.

The item only returns the last answer. This is considered a drawback, because earlier (wrong) answers, which could reveal important information, are lost. This situation is caused by using the Items-D specification, which does not provide answer histories in the item result files.

**Observation of students using the system**

In the first hour, students really seemed motivated in obtaining high scores. Although especially with the first test it was very easy just to give two random answers to obtain the solution, students did not act like that. They really tried to
solve the problems given to them. The methods of solving differed: some eliminated impossible answers and chose a likely answer without calculating very much, many others calculated the answers accurately. Sometimes there was a discussion between two students, but that can only be encouraged in a training environment. Usually the hint had the effect of putting the student back on the right trail. When a student had entered two incorrect answers, the subsequent solution was read really carefully to see what had gone wrong, and now and then there was a sound of recognition. After a correct answer, the students usually skipped through the solution very quickly, sometimes a little annoyed that it was shown to them. In the adaptive session it was interesting to see what happened when the system chose a comparable item after an erroneously answered item. In many cases this item was done correctly.

A loss of interest after about an hour could be seen very clearly during the session. Some started walking around, getting coffee, making jokes or giving nonsense answers just to see what happens. This observation corresponds with the feedback received from the students as described in section 7.3.2.

7.3.4 Examples of adaptation paths

Figure 7.7 shows four examples of how the system has adapted the level in the sessions. The numbers at the x-axis show the subject number and the stage (i.e., 1 means the stage to get the state to probably, 2 means the stage to get the state to strong. The bars show the answer score: 1 means correct, 0 means incorrect, 0.7 means correct in two attempts.

In the first example, the first stage was finished without mistakes in three items. The influence of a mistake can be seen in the second stage. The desired level drops and it takes a few items to recover. The second example shows a situation in which the mistakes occurred in the first stage. In the second stage, it took only two items to conclude mastery, because two items of difficulty 0.8 were answered correctly.

The third example shows a situation in which the process could better have stopped. It did not do that because it could not find any items with a difficulty level smaller than 0.2; therefore the estimation did not get below the threshold. A way to make certain this does not happen in the future is to make a special item of a very low difficulty, which, if done incorrectly, gives almost certainty that the student will have to study first. If there had been an item of difficulty 0.05 which would have been answered incorrectly, the process would have stopped. The fourth example shows the effect of the small number of items best. In almost none of the cases was the system able to find an item close enough to the desired level. For example, there was no item of difficulty 0.3. This resulted in still fairly large jumps.
In conclusion, the system uses the available resources quite well, but it needs a larger item bank to prevent some undesirable effects.

7.4 Conclusions

Although the number of participating students in the test was low, the results show clear trends that the system meets a need. The goal to activate the students definitely succeeded. The path that the system follows through the items does not give unexpected results.

Quantitative analyses of the session results give only moderate correlations with examination results. This can partly be explained by the item selection strategy (continuing on a different subject when a certain level has been reached), partly by the small size of the item bank, and partly by the small number of participating students. Further, students tend to study in peaks, just before the examinations. Their "forgetting factor" is probably quite high. For a more accurate result, the test should be repeated after enough items have been produced, with more participating students.
Experiments and results

It should be taken into account that for a training system, other priorities apply than for a testing system. With testing, accuracy of the estimates is very important because there are major consequences for the student; an inaccurate estimate could mean that a competent student fails or an incompetent student passes. With training, accuracy is less important than the act of training (with improvement as aim).

It cannot yet be concluded whether the system saves the teacher’s time. For the test it showed that making items takes a lot of time. However, if the items are available and the system is running autonomously, this will most likely be paid back.

One of the points lacking in the system is a recommendation. The student still has to see for himself whether or not he/she needs extra training for the exam. In some cases this can be quite unclear; a direct translation from the subject scores to an examination score cannot be done. In other words, the proficiency estimates, together with the time spent and the item history will have to be translated into a more readable and understandable form. A possible implementation is a knowledge-based subsystem with expertise both on subject matter pointers and on evaluation of the results. Likely conclusions would be:

- You seem to handle problems on X quite well, but you can use some more practice, because you proceed too slowly for a good examination result.
- Y seemed to be a problem area. You are advised to study chapter x,y of the book again and to do another training session after that.
- Although you answered even some difficult questions on Z correctly, the subject has probably not been mastered very well, because there were also a lot of mistakes.

Another (probably more easily implementable) method is providing trial examinations with a representative set of items on examination difficulty.

Very important questions are: does the system enhance the effectiveness of studying, and is the system a possibility for putting the students to work and keeping them at work? The test sessions give optimistic results concerning student activation. The question remains as to whether this is a result of the novelty of the system, or of the system itself. Will an improvement in examination grades occur when the system is available? Or will the students need less time to reach the same level of mastery? Will the students comprehend and retain the course material better by practising problem solving? Will the students remain enthusiastic about the system? These questions cannot be answered with a single, small scale experiment. New tests will have to be performed to see the long-term effects of the system.
Conclusions and recommendations

This thesis has described the considerations, requirements, design and first tests of a generally applicable training system in engineering education. First the research outcomes are presented. Next, there is a research reflection on the research as basis for the recommendations. Then follow technical, organisational and didactic recommendations.

8.1 Research outcomes

Effective learning is stimulated by active student participation, which does not occur enough in large group lectures. In engineering education, training is important to acquire the needed skills. Since the teachers only have a limited amount of time to spend on individual students, a computer-based training program can offer a valuable contribution. To keep the students challenged, the training should be adaptive to the student's mastery level. Therefore, either student control (in which the student takes care of the adaptation himself) or adaptive program control can be used. Training based on the latter has the advantage of efficiency: results are not necessarily better, but they are often reached in a shorter time. Student control usually positively influences the students' motivation, but it is not always effective for low achievers. With the possible later incorporation of student advisory control (advice based on an adaptive mechanism, but the advice not enforced on the students) it was decided to focus on adaptive program control.

An important aspect of the research is implementability in education. A factor to take into account is that teachers usually have many items at their disposal. Further, by giving the teachers the possibility to implement the items without needing a pro-
grammer, the implementation cost can be minimised. A training system should allow the use of own items and easy implementation by non-programmers.

For a generally applicable system for pre-examination training, traditional CAI, (i.e., CAI where content and paths are hard-coded into the program) is too rigid. Some intelligent tutoring systems, which are flexible, have been made, but they are too domain dependent. Standard item banking systems usually ask too much teacher activity after implementation, or pose too many restrictions for item implementation. Adaptive item banking seems a viable solution. In this context, "adaptive" means that the selection of a new item is influenced by the answers to previous items. A restriction is that engineering domains usually need an adaptation strategy which allows for using multiple skills or subjects.

Several, initially thought promising, AI techniques have been examined on their usability in the system, in particular machine learning techniques. It appeared that machine learning techniques are not the most likely candidates for use in the system. A blackboard architecture could be used very well as a basis for the system. It consists of independent agents cooperating to solve a problem, thereby acting on a shared data structure.

A prototype for a general training shell has been realised in C++. The system is based upon the idea that it is possible to create a taxonomy of subjects for the target course. For each subject a unidimensional subject model is used. Item models are reflections of these models, containing information about the subject(s) and parameters needed by the unidimensional models.

For instantiation of the models, simple methods have been used. The course structure is modelled as an and-graph, i.e. all child nodes are prerequisites for a node. For the unidimensional proficiency model, a method based on IRT was chosen. The item model is a standard 1-PL model, but the update process uses a heuristic algorithm which does not have the large jumps that can occur in standard IRT. The method appeared to be satisfactory.

For the easier implementation of items, a toolbox for control theory related graphs was built; many of these graphs are also applicable in other domains, specifically in electrical engineering. An Items-D type interface was realised for Taiga. With these extensions, Taiga is a usable environment for writing static items. Dynamic items are better made in another environment. An example of a dynamic item template for the connection between time responses and pole-zero configurations was built using Pascal and the transformation package TRIP.

The prototype of the adaptive system was tested in a group of 29 students. The system behaves as intended; the shell is usable, and the current implementation gives
reasonable results. The group of students and the number of items were too small to make founded judgements about the reliability of the methods, but it appeared that students who perform better at the examination also tend to perform better in the training with two of the four subjects. The most likely cause of trouble with the other two subjects appeared to be that some items contained errors. From this, it can be concluded that it is very important that the items be correct, because the consequences can propagate via the adaptation.

The students have appeared to be willing to participate in tests and evaluation sessions (albeit not completely voluntarily as the tests were included in an obligatory practical), and did not hesitate in giving critical remarks or showing appreciation. The students' general opinion about the system was positive. Most criticism was focused on item details and on the lack of influence they had in choosing the subject. Further, the ability summary was found to be too limited.

With the current system, the main aims as stated in the introduction have been reached. Teachers can implement and maintain items themselves while having graph and diagram display facilities available. It is expected that the system will also save time for the teachers and instructors, but this has not yet been shown. Initially it will still take a lot of time to implement items, but later on, when items are available and the system is running, less attention will be needed.

The system is domain independent to the extent that it can be used for other courses if the structure of those courses (the taxonomy of subjects) is available. The claim of general applicability of the system has been made plausible: although the system was initially aimed at a course in control theory, it appeared possible to use the system for a course in electronics without software modifications.

It is also possible to adapt the system to another teacher's preferences, and to use any implementation environment. Further, the system can be used both stand-alone (e.g., for use at home) and in a network.

8.2 Project reflection

This section describes the path of the research. It is used in later sections for recommendations with respect to future projects in this field of research.

The project can be split into two sections: in the first two years, starting in the autumn of 1990, the focus was on a course on systems and control engineering. Initially the idea was to use a management system which was designed concurrently,
Conclusions and recommendations

so the research was focused on the design of facilities for the easy implementation of items, and on implementation of items themselves. In-line simulation was also applied in some items.

In the autumn of 1992, three major events took place. First, the electrical engineering curriculum was restructured. As a result, the course the system was aimed at was removed from the second year; a replacement course is available in the third year. This made field tests impossible, because the course was not given for a year. Further, as the new course is voluntary, fewer people are taking it; using a training system would save less time for the teacher than in the old situation, because the same teacher effort would be needed for a smaller number of students.

Second, the design of the management system appeared to give problems. There were some personnel changes in the unit responsible for implementation, resulting in a shift of focus of interest. The proper implementation of the management system seemed to become uncertain. Further, the management system was mainly aimed at tests using a limited and pre-specified number of items. In test sessions, this rigidity had already appeared to be a disadvantage. It was thought that an adaptive session manager would be more appropriate. The developers of the management system made it clear that implementing such a manager would not be possible for them.

Third, the Electronics Laboratory also started a project to develop a training system for a first-year course in electronics. It was decided to join forces and to make a generally applicable training system. The results of the first two years were applied at the Electronics Laboratory to implement items, and the focus of our research was shifted to the implementation of an adaptive session manager. Items for both the non-adaptive and the adaptive system would have to be exchangeable.

In September 1993, a student acceptance test of a non-adaptive training system was performed with 20 items. The main goal was to investigate the students’ opinions about the items. The results were encouraging enough to continue the research using the same item implementation environment. In the spring of 1994, the adaptive session manager was finished. The item set of the previous test plus a number of new items were used to make an item bank of 48 items. The second student test, mainly meant to evaluate the adaptive system, and to a less extent the items, was done in June 1994.
8.3 Recommendations

8.3.1 Technical recommendations

The prototype has been tested on a fairly small number of students. Larger scale tests will be necessary before making founded judgements about reliability. For these tests to be more useful, more items will have to be implemented.

At the moment, the system relies fully on the teacher’s estimates for item parameters. Analysis programs can be implemented, able to estimate these parameters from the student responses, provided enough response patterns are available. As this is a real classification task, a statistical method can be applied, or a machine learning algorithm. With such an extension, the parameters of misclassified items will converge to a measured value. Of course, the teacher must always keep an eye on the system; for example, it is suspicious if an item’s level of difficulty appears to become very low while the teacher had expected a high value.

The system is not yet able to give study advice (i.e., a discussion of needed attention to subjects and, for example, literature pointers). The session survey gives an indication, but it still has to be interpreted by the student. A knowledge-based system that translates the survey into study advice would be a welcome extension.

Although reasonable successes have been booked, the system is still in the prototype stage. Before it can be applied on a large scale, it must be brushed up. For example, at the moment, item characteristics, course model and initial subject models must be entered in a rather primitive way. A program supporting this process could provide a great service. Further, translation phases, to be able to use data bases for other models, could be valuable.

8.3.2 Organisational recommendations

When incorporating computer support in education, it is important that it is organised properly. This section gives suggestions on some aspects of how to make the system available to students, what to take into account when designing a system, and how to organise projects. Suggestions for the incorporation of training systems in the curriculum are given with the educational recommendations in section 8.3.3.

A first question to be addressed is: in which way must the system be available? A big advantage of using a computer network at the university department is the possible storage of results for later analysis. This method was used in the test sessions. When the system runs on stand-alone machines with the students at home, the results are not available. However, students often prefer to study and practice at
Conclusions and recommendations

home. A possible solution to this dilemma is use of the Internet. Students with a computer and a fast modem can log in to the university net, which gives access to Internet as well. All available computers in the department are already connected to the Internet. By making the system available via the Internet, it is possible to maintain statistical information, but also to offer the student continuous accessibility to the system, both at home and at the department. Until recently, using the Internet implied not being able to use graphics, especially when using a modem. With the current fast modems and the introduction of the World Wide Web (WWW), usage of graphics is also feasible. There are possibilities of using the WWW as a medium for the system; these are discussed in Appendix C. At the moment, these possibilities are not used very widely; the services are mainly used for presentation, not for interactive applications. An issue to be taken into account is network capacity.

A second issue is the organisational aspect with system design. To incorporate training systems in engineering, the system should not be too rigid to support independence. For example, if teacher-defined sessions with limited sets of items are used, the sessions will have to be ‘refreshed’ regularly by the teacher, or else the items will become known, causing, at least, biases in estimation. Therefore, the teacher would still have to spend much time on routine work. For a training system to be efficient for the teacher, it should be possible to run without teacher interference (other than modifying the item bank).

Further, systems should be portable and reusable in other courses in order to prevent every interested teacher from having to “re-invent the wheel”.

Projects should have a larger basis than one course. A joint approach (for instance, on departmental level) has more chance of success because it does not depend on the enthusiasm of one individual. Further, course (or teacher) dependencies can be spotted earlier if a system is applied in more courses. If a framework is available, it is much more easily applicable by teachers, because then the main skill needed (to make items) is their area of expertise. However, the project goals should not be too broad. For example, projects to make a general system for testing and training, both interactive and using paper and pencil, both adaptive and non-adaptive is likely to deliver a large program which is unwieldy to use for the teachers.

With joint projects, clear agreements must be reached in an early stage of the project, to prevent conflicting points of view from making a project fail.
8.3 Recommendations

8.3.3 Educational recommendations

The system is designed for courses that can be divided into a taxonomy of subjects with possible prerequisite relations, as is very often the case in engineering. It is especially useful for courses that satisfy the following conditions:

- Practising (e.g., for examination training) in the form of problem solving is deemed important for mastery of the subject matter.
- The problems can be offered in a closed format (numerical answers, multiple choice, etc.) so that the answers can be interpreted easily by a machine.
- The course is aimed at larger groups of students. The system will especially save time for teachers when used with large scale courses. Also for medium scale courses it can be used, but then more in order to give the students an individual guidance than to save time for the teacher.

Concerning adaptation, the test sessions showed that students do not have major objections to adaptive system control, provided that they can influence at least the subject to be trained. Complete student control was not found very desirable: many students said they appreciated the guidance. A strategy that takes these considerations into account is student advisory control (i.e., give advice based on an adaptive mechanism, but leave the decision to the student). This can be implemented in several degrees of student control, for example by giving several options as well as a preferred action, or by suggesting an action but allowing the student to choose item characteristics. The negative effect is that the student influences the optimal choice of items according to the adaptation scheme, but the positive effect is that it will be more readily accepted by the students. The impression of having control seems important.

With some courses, a testing system is used to activate the students. They have to do a test regularly (e.g., every week), and the results of these tests are used to decide whether the student is admitted to the examination. Although, undeniably, this activates the students in that course [Dijkman & Staal, 1991], it could well be at the expense of their doing less for other courses. Students seem to reserve a certain amount of time for studying, and if one course needs more time, less time is given to another. Therefore the introduction of such systems can have a positive effect on the course, but a negative effect on courses given in parallel. Consequently, testing systems may not always be the most desirable option.

However, students do show interest in practising; they know it is a good way to prepare for the examination. Previously written examinations are available with solutions, and they are bought regularly. The problem (and the students themselves
Conclusions and recommendations

admit that as well) is that the solutions are “too available”. It is too easy to look at the solution, and to assume, by following the line of reasoning, that you are able to find the answer yourself. A training system that does not give the solution before an attempt was made to solve the problem hopefully gives just that extra push to make the student more active. Further, when the training is voluntary, the risk of taking away study time from other courses is much less. It is advisable to include at least one training session quite early in the course, to introduce the students to the system. In addition, it should be available continuously for voluntary use.
References

AAMODT, 1994

AKSIT ET AL., 1992
Aksit, M. Bergmans, L & Bosch, J., Object-Oriented Design. PAO informatica course, University of Twente, Enschede (NL), 1992.

BAIN, 1986

BELL & FEIFER, 1992

BIERMAN, 1991
Bierman, D.J., "To be intelligent or not to be intelligent: Is that the question?". Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, pp. 25-34. Lausanne (CH), 1991.

VAN DEN BOSCH & BUTLER, 1990
References

BRAKTO, 1994

BROPHY & ALLEMAN, 1991

TEN BRUGGENCATE ET AL., 1993

BISON, 1994

VAN BRUGGEN & VAN DE VEN, 1989

CANDY, 1986

CHUNG & REIGELUTH, 1992

COHEN, 1981

COMPUERSE, 1990
CompuServe Inc., Graphics Interchange Format Programming Reference, Columbus (Ohio, USA), 1990.

COSTA, 1992
DIJKMAN & STAAL, 1991

DOIGNON & FALMAGNE, 1985

DORF, 1989

ESKENASI ET AL., 1993

ESSENIEUS ET AL., 1993

ESSENIEUS, 1994

FRASER ET AL., 1987

FRICK, 1989
FRICK, 1990

GLASER, 1977

GLASER, 1982

GODFREY & STERLING, 1982
Godfrey, D. & Sterling, S., The elements of CAL. Reston publishing company, Reston (Virginia, USA), 1982.

GOLDSTEIN, 1982

GUSKEY, 1988

HAASDUK ET AL., 1994

HASSELERHARM & LEEMKUIL, 1990
HONDERD, 1989
Honderd, G., *Syllabus bij de colleges Systemen en Regeltechniek (184 I & 184 II)*. Syllabus nr. Q86.063, Delft University of Technology, Department of Electrical Engineering, Control Laboratory, Delft (NL), 1989.

HONDERD & JONGKIND, 1990

HUMPERT, 1990

JAGANNATHAN ET AL., 1989

JANSEN ET AL., 1991

JOCHEMS, 1990

KAELBLING, 1994

KERAVNOU, 1990

KOHNERT & JANKE, 1990.
Kohnert, K. & Janke, G., “Knowledge representation with and-or-graphs: comparing the approach of Doignon & Falmagne with the ABSYNT-diagnostics”.

177

**Kubat, 1992**

**Kurland & Kurland, 1987**

**Kyllonen & Shute, 1989**

**Lee, 1992**

**Lesgold et al., 1989**

**Lysakowski & Walberg, 1982**

**Michalski, 1986**
MICHALSKI, 1993

MINSKY, 1962

PULLEN & MERCER, 1988

RASCH, 1960
Rasch, G., Probabilistic models for some intelligence and attainment tests. Denmarks Paedagogiske Institut, Copenhagen (DK, USA), 1960.

VAN DE REE, 1994

SCHANK & SLADE, 1991

SCHOONMAN, 1989

SLEEMAN & BROWN, 1982

SUPPES ET AL., 1968

TACONIS & FERGUSON-HESSLER, 1994
Taconis, R. & Ferguson-Hessler, M., “Het belang van probleemoplossen voor het onderwijs in technische en exacte vakken” (The importance of problem

**Tennyson & Christensen, 1988**


**Wainer, 1990**


**Walberg, 1984**


**Walberg, 1988**


**Wald, 1947**


**Wenger, 1987**


**Wijnen, 1992**


**Winkels & Breuker, 1992**

Wissenburgh, 1991
Wissenburgh, C., Inleiding in de Elektronica. Een ontwerpgerichte benadering
(Introduction to Electronics. A design oriented approach). Delftse Uitgevers

Software

Inigo, 1994
Educational Centre University of Twente, Inigo Version 2.1. Enschede (NL),
1994.

Taiga, 1987
Educational Centre University of Twente, Taiga. Enschede (NL), 1987.
Abbreviations

ACBT  Adaptive Computer-based Training
AI    Artificial Intelligence
ANN   Artificial neural net
CAI   Computer-Assisted Instruction
CAT   Computerised Adaptive Testing
CBR   Case-based reasoning
CBT   Computer-based Training
CT²   Control Toolbox for Taiga
DFD   Dataflow Diagram
GA    Genetic Algorithm
GIF   Graphic Interchange Format
GKS   Graphical Kernel System
GT²   General Toolbox for Taiga
HTML  HyperText Mark-up Language
I²T   Items-D type Interface for Taiga
ICAI  Intelligent Computer-Assisted Instruction
ICC   Item Characteristic Curve
ITS   Intelligent Tutoring System
IRT   Item Response Theory
LZW   Lempel-Ziv-Welch
ML    Machine Learning
PI    Programmed Instruction
PP    Psycho-pedagogical
SC    Structure Chart
SPRT  Sequential Probability Ratio Test
TRIP  TRansformation and Identification Program
WWW   World Wide Web
Appendix A

Format specifications

In this appendix, the format specifications of the interface files and other data file standards are explained in more detail. Section A.1 describes the item information file and section A.2 describes the item result file, both according to the Items-D interface specification. In Section A.3, the Taiga text format is described. The syntax definition uses a derivative of the Backus-Naur form:

- \{entry\} optional entry (0 or 1 time).
- \{entry\} * optional multiple entry (0 or more times).
- \[entry\] * n exactly n times the entry.
- entry1 | entry2 entry1 or entry2.
- (entry1 entry2) treat multiple entries as one.

Italicized entries denote values. There are 6 types of values:

- Integer values, if the name starts with number-of.
- Real values, if the name starts with value-of.
- Real values with exception, if the name starts with value*-of. With this type, an asterisk (*) can be used as the value, denoting a special case.
- Character values, if the name starts with char-of.
- Text values, if the name starts with text-of.
- Name values, if the name starts with name-of. Many environments limit the possibilities for naming variables. This should be taken into account.

Bold text should be treated as literal, with the standard C notation for special characters such as newline (\n). A definition is given for entries in the default font.
A.1 The item information file DATA.TDL

The item information file is used to pass information from the session manager to the item.

A.1.1 Syntax

item-information-file ::=  
   {parameter-section} {answer-section} {miscellaneous-section}

parameter-section ::=  
   BEG_PAR
   number-of-params \n   [parameter-specification] * number-of-params  
   END_PAR

parameter-specification ::=  
   name-of-parameter , value-of-parameter \ text-of-parameter

answer-section ::=  
   BEG_ANT\n   value-of-itemscore \n   number-of-answers \n   [answer-specification] * number-of-answers  
   END_ANT\n
answer-specification ::=  
   value-of-score \n   number-of-answers \n   name-of-answer , number-of-answertype , value-of-answerscore \n   number-of-anticipated-answers \n   [anticipated-answer-part] * number-of-anticipated-answers

anticipated-answer-part ::=  
   value*-of-answer , value*-of-minimum , value*-of-maximum ,  
   value*-of-score \n   feedback-part

feedback-part ::=  
   BEG.TXT\n   text-of-feedback  
   END.TXT\n
A.1 The item information file DATA.TDL

miscellaneous_section ::= 

BEG_DIV
  {global-section} {local-section}* {file-section}*
END_DIV

global-section ::= 

GLOBALAL
  [ name-of-score-influencer = percentage ]*

local-section ::= 

LOKAAL name-of-answer
  [ name-of-score-influencer = percentage ]*

percentage ::= 

( * value-of-percentage % ) | -rest

file-section ::= 

FILE name-of-file number-of-condition
  {file-contents}*

file-contents ::= 

char-of-file | %name-of-variable%

A.1.2 Semantics

The parameter section contains the number of parameters, followed by the parameter specifications. Parameters can be of any of the types real, integer, text (max. 80 characters) or Boolean (N or F becomes False; Y, J or T becomes True). Choice of parameter names can be restricted by the item environment.

The answer section gives a specification of the obtainable scores and possible answers to the (sub)question(s) in the item. Answer types can be numerical (code 1), 2-9 choice, (codes 2-9), classification (code 10) or textual (code 11). For the anticipated answers, an interval is specified. If an asterisk is used for value*-of-minimum, the lowest possible value is applied (for example, with normal floating points: 1e-38), if it is used for value*-of-maximum, the highest possible value is applied. For value*-of-answer, an asterisk means any value. So an anticipated answer specification with all asterisks can be used as a safety net for unanticipated answers. The list is processed from top to bottom, so such a safety net should always be the final entry. For every answer specification entry, no more than 20 characters may be used.

Normally, the alternative score for the correct answer is equal to the answer score. With suboptimal answers, the score can be less than the answer score. A score of 0 means that the corresponding answer alternative is not correct. If no score should be
assigned to the answer (e.g., with intermediate results), the right answer is specified by an asterisk as the score.

The feedback text is a line of text to be displayed if the student answer matches the alternative. This can be used to give a hint in the right direction when an incorrect answer was given.

The miscellaneous section specifies other information about the item. The global subsection defines percentages to be subtracted from the maximum score on certain occasions, for example if the student gives an incorrect answer or asks for help. These are applied globally, for every question in the item. A local override can be generated by means of a local subsection for one question only.

Finally, the file subsection specifies files that should be available on a certain condition. The condition is specified by an integer value. The meaning of this value is item environment dependent. This feature can be used to generate data files depending on parameter values, which can be inserted by using the variable name surrounded by percentage-signs.

A.1.3 Example

BEG_PAR
3
Scale, 1e-3
Length, 1
Width, 4
END_PAR

Start Parameters-part
There are 3 parameters:
Scale, with value 0.001
Length, with value 1
Width, with value 4
End of parameters part

BEG_ANT
50
2
Contour, 1, 25
3
1e-2, 9.9e-3, 1.01e-2, 25
4e-6, 3.99e-6, 4.01e-6, 0
END_ANT

Start of answer part
Maximum achievable score is 50 points
Two questions in the item
First is Contour, type 1 (numerical), max.score 25
3 foreseen answers:
Answer 1e-2 (correct) gives 25 points
Answer 4e-6 (Area calculated) gives 0 points
This alternative also defines a hint text

Didn't you calculate the surface?

All other answers (incorrect) also give 0 points

Area, 1, 25
3
Second answer is Area, same type and max.score
3 foreseen answers:
1e-2, 9.9e-3, 1.01e-2, 0
BEG_TXT
Didn't you calculate the contour?
END_TXT
4e-6, 3.99e-6, 4.01e-6, 25
*, *, *, 0
END_ANT

Answer 1e-2 (Contour calculated) gives 0 points

Answer 4e-6 (Correct) gives 25 points
All other answers (incorrect) also give 0 points

End of answer part

BEG_DIV
GLOBAAL
1e_fout = -50%
2e_fout = -rest
hulp = -10%

Start of miscellaneous part
Start of global definitions
first error gives a 50% reduction in score
second error reduces score to 0
Asking for a hint costs 10%

LOKAAL Contour
hulp = -20%

Local overrides for problem Contour:
Asking for a hint costs 20%

FILE AUTOEX.TRP 2
rss
0
1
%Length%
1
0
do sst sy
mway ty stapresp
exit

End of miscellaneous part

A.2 The item result file RESULT.TDL

The item result file is used to return information from the item to the session manager. Also intermediate information (when answering of an item is temporarily aborted, with the intention of continuing later) can be stored.

A.2.1 Syntax

item-result-file ::= 
item-section answer-section {miscellaneous-section}
item-section ::= 
    BEG_VR\n    VRAAGSTATUS= number-of-itemstatus \n    VRAAGSCORE= value-of-itemscore \n    END_VR\n
answer-section ::= 
    BEG_ANT\n    { answer-specification }* 
    END_ANT\n
answer-specification ::= 
    ANTWOORDNAAM= name-of-answer \n    STATUS= number-of-status \n    ANTWOORD= value-of-answer | text-of-answer \n    DEELSEORE= value-of-score \n    { AANTAL_FOUTEN= number-of-errors \n    { name-of-score-influencer = number-of-score-influencer \n    }

miscellaneous_section ::= 
    BEG_DIV\n    {special-section}* 
    END_DIV\n
special-section ::= 
    name-of-special-entry = 
    value-of-special-entry | text-of-special-entry \n
A.2.2 Semantics

The item section gives a summary of the result. It contains the item status (0 = not done, 1 = interrupted, 2 = finished) and the total item score.
The answer section gives an overview of the result. Per question it gives at least the name, status, given answer, and score. It can also contain the number of errors the student made and an indication as to whether other score influencing factors (e.g., asking help) have been applied. The entered number is a binary value (0 for not used, 1 for used).

The miscellaneous section is meant for intermediate results, to be used if it is allowed to interrupt items and resume later. The information contained in this section is environment dependent.
A.3   External text file format of the General Toolbox

A.2.3   Example

This example shows a possible result after running an item with the example data file of section A.1.3.

```
BEG_VR
VRAAGSTATUS=2
VRAAGSCORE=45
END_VR

BEG_ANT
ANTWOORDNAAM = Contour
STATUS = 2
ANTWOORD = 1e-2
DEELSCORE = 20
AANTAL_FOUTEN = 0
HULP = 1
```

Start of item section
Item has been finished (status 2)
The student obtained 45 points
End of item section

Start of Answer section
Concerning question Contour:
Question was finished
The correct answer, but
Help was asked, so 20% reduction
No errors made
Help asked

```
ANTWOORDNAAM = Area
STATUS = 2
ANTWOORD = 4e-6
DEELSCORE = 25
AANTAL_FOUTEN = 0
HULP = 0
END_ANT
```

Concerning question Area:
Also finished
Correct answer, and
Maximum partial score
No errors made
No help asked
End of Answer section

A.3   External text file format of the General Toolbox

A.3.1   Description

The text files used in the external text display facility consist of one or more pages of text in Taiga format. This is the format that Taiga uses in the development system for saving text windows. It is an ASCII format with escape sequences for font changes, underlining, reverse video and colour changes. If no special features are used, the text is plain ASCII. Escape sequences have the following format:

```
char-of-escape char-of-colour number-of-attribute number-of-font
```
In this sequence, char-of-escape is the normal ASCII escape code (27 decimal); char-of-colour is the colour code as defined in Taiga (see table A.1).

**Table A.1. Colour codes in Taiga**

<table>
<thead>
<tr>
<th>Code</th>
<th>Colour</th>
<th>Code</th>
<th>Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Black</td>
<td>H</td>
<td>Transparent</td>
</tr>
<tr>
<td>R</td>
<td>Red</td>
<td>I</td>
<td>Light green</td>
</tr>
<tr>
<td>G</td>
<td>Green</td>
<td>J</td>
<td>Dark blue</td>
</tr>
<tr>
<td>B</td>
<td>Blue</td>
<td>K</td>
<td>Dark red</td>
</tr>
<tr>
<td>C</td>
<td>Cyan</td>
<td>S</td>
<td>Light red</td>
</tr>
<tr>
<td>Y</td>
<td>Yellow</td>
<td>T</td>
<td>Light yellow</td>
</tr>
<tr>
<td>M</td>
<td>Magenta</td>
<td>U</td>
<td>dark green</td>
</tr>
<tr>
<td>W</td>
<td>White</td>
<td>V</td>
<td>Grey</td>
</tr>
</tbody>
</table>

The code number-of-attribute is an addition of the codes for underline (4) and reverse video (1). Finally, the font code number-of-font is 0, 1 or 2, giving the default font, and alternative fonts 1 and 2, respectively. These alternative fonts are defined in the files altset1.scs and altset2.scs.

An obvious section separator is a form feed character (ASCII 12). Text between the form feed character and the next new line will be discarded, therefore the line can be used for comments.

**A.3.2 Discussion**

Given that the texts will mostly be typed by people concentrating on item content, the format must be easy to use. The format allows the possibility to use the Taiga character sets, change colours and use other facilities such as underline and reverse video in the format the Taiga production environment uses when saving text windows. Further, a clear separator is defined to distinguish between two different text sections in one file.

The advantages of using this format are obvious: it is a simple format which can be edited with any text editor, and it is easy to convert existing Taiga items by simply saving the texts. A drawback is that, especially when using formulae with special characters, the text is not easy to read, because special characters are written in the normal font, surrounded by escape codes. If this appears to be a major drawback, a simple dedicated editor could improve the ease of use. It is also possible to use the Taiga text editor, save the texts to files, and integrate the files using an editor.
Appendix B

The questionnaire

This appendix contains the questionnaire that was handed out after the second student experiment. As the original questionnaire was in Dutch, an English translation is also included.

In the following sections, the original Dutch version, as handed to the students, is given, and the English translation of it. In the Netherlands, exam marks range from 1 to 10, 1 being extremely bad, 10 being excellent.

Following the questionnaire text, the raw results of the questionnaire correlation analysis are given.
**B.1 The original Dutch version**

### Vragenlijst oefeningen bij Elektronica practicum

Deze vragenlijst bestaat uit 25 meervoudvragen. Antwoorden kunnen worden aangegeven door omcirkelen of aankruisen op de eronder staande balk. Na de vragen is ruimte voor eventuele opmerkingen. De resultaten van deze vragenlijst worden slechts gebruikt ter evaluatie van de gebruikte programmatuur.

**Voorbeeld:**

<table>
<thead>
<tr>
<th>a</th>
<th>Het aantal in deze enquête gestelde vragen is</th>
<th>≤5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>&gt;30</th>
</tr>
</thead>
</table>

**Studienummer: [Insert]**

#### Algemeen (tenzij anders vermeld m.b.t. het vak Elektronica)

<table>
<thead>
<tr>
<th>1</th>
<th>Mijn vooropleiding is</th>
<th>VWO</th>
<th>HBO</th>
<th>HBO-P</th>
<th>anders</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Mijn tentamencijfer Elektronica (et04-10) was</td>
<td>≤4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Mijn gemiddelde tentamencijfer voor Elektrische circuits 1/2 (et08-10) was</td>
<td>≤4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>De tijd die ik tot nu toe besteed heb aan zelfstudie m.b.t. de collegeles is in uren ca.</td>
<td>≤5</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>Het aantal colleges dat ik gevolgd heb is te ontschrijven als</td>
<td>geen</td>
<td>de helft</td>
<td>alle</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Het aantal door mij gevolgde instructie-uren is</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

#### Oefening algemeen

| 7 | De oefening sluit goed aan bij de behandelde leerstof | oneens | eens |
| 8 | Tijdens het doorlopen van de oefening was het duidelijk wat er van mij verwacht werd | oneens | eens |
| 9 | Tijdens het gebruik van programma kwamen de programma-acties logisch op mij over | oneens | eens |
| 10 | Het oproepen van opgaven gaat voldoende snel | oneens | eens |
| 11 | Het tijdens de sessie gegeven overzicht geeft een goed beeld van mijn prestaties tijdens de sessie | oneens | eens |

#### Vraagstukken

<p>| 12 | Ik had snel door hoe je het juiste antwoord moest aangeven | oneens | eens |
| 13 | De lay-out van de vraagstukken vind ik overzichtelijk | oneens | eens |
| 14 | De hints en opmerkingen na controle van het antwoord zijn duidelijk | oneens | eens |
| 15 | De uitwerkingen van de vraagstukken vind ik duidelijk | oneens | eens |</p>
<table>
<thead>
<tr>
<th></th>
<th>Doorgaans prefereer ik bij dit soort oefeningen een uitgebreide uitwerking boven een beknopte.</th>
<th>oneens</th>
<th>eens</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>De schermopbouw vind ik snel genoeg.</td>
<td>oneens</td>
<td>eens</td>
</tr>
<tr>
<td></td>
<td><strong>Opeenvolging van de vragen</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>De keuze van onderwerp en niveau kwam telkens logisch op mij over.</td>
<td>oneens</td>
<td>eens</td>
</tr>
<tr>
<td>19</td>
<td>Ik wil zelf kunnen aangeven welk deel van de stof ik wil oefenen.</td>
<td>oneens</td>
<td>eens</td>
</tr>
<tr>
<td>20</td>
<td>Er is een systeem gebouwd dat een volgende vraag selecteert uit een vragenbank op basis van getoonde resultaten. Dit principe bevalt mij goed.</td>
<td>oneens</td>
<td>eens</td>
</tr>
<tr>
<td></td>
<td><strong>Mijn motieven voor de keuze in vraag 20 zijn:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Ik vind het handig als het systeem voor mij de moeilijkheidsgraad afstemt op mijn prestaties.</td>
<td>oneens</td>
<td>eens</td>
</tr>
<tr>
<td>22</td>
<td>Ik vind het belangrijk om zelf te kunnen bepalen welke vraag ik als volgende ga beantwoorden.</td>
<td>oneens</td>
<td>eens</td>
</tr>
<tr>
<td>23</td>
<td>Ik vind het belangrijk om bij verschillende sessies verschillende vragen te krijgen.</td>
<td>oneens</td>
<td>eens</td>
</tr>
<tr>
<td></td>
<td><strong>Oordeel</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Mijn algemene oordeel over deze wijze van oefenen is</td>
<td>negatief</td>
<td>positief</td>
</tr>
<tr>
<td>25</td>
<td>Met enkele modificaties zou deze programmnatuur ook geschikt gemaakt kunnen worden voor ten-</td>
<td>negatief</td>
<td>positief</td>
</tr>
<tr>
<td></td>
<td>taminering. Mijn houding ten opzichte hiervan is</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Opmerkingen en suggesties ter verbetering:
### B.2 The English translation

**Questionnaire "Exercises with Electronics laboratory"**

This questionnaire consists of 25 multiple choice questions. Answers can be entered by encircling or ticking on the bar below the answer options. After the questions there is room for possible remarks. The results of this questionnaire are only used for evaluation of the used software.

**Example:**

<table>
<thead>
<tr>
<th>a</th>
<th>The number of questions asked in this questionnaire is</th>
<th>≤5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>&gt;30</th>
</tr>
</thead>
</table>

**Student number: ____________**

**General** (unless noted otherwise with respect to the course Electronics)

<table>
<thead>
<tr>
<th>1</th>
<th>My pre-university education is</th>
<th>VWO</th>
<th>HBO</th>
<th>HBO-P different</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>My exam mark for Electronics (et04-10) was</td>
<td>≤4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>My average exam mark for Electrical circuits 1/2 (et08-10) was</td>
<td>≤4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>The time I have spent on home study regarding the course material is in hours approximately</td>
<td>≤5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>The number of course lectures I followed can be described as</td>
<td>none</td>
<td>half</td>
<td>all</td>
</tr>
<tr>
<td>6</td>
<td>The number of instruction hours I followed is</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

**Exercise in general**

| 7 | The exercise fits well with the treated course matter | disagreed | agreed |
| 8 | During the exercise it was clear what was expected from me | disagreed | agreed |
| 9 | During the use of the program the program actions came across logically to me | disagreed | agreed |
| 10 | Execution of items is satisfactorily fast. | disagreed | agreed |
| 11 | The survey given between to items gives a good impression of my performance during the session | disagreed | agreed |

**Items**

<p>| 12 | I found out quickly how the correct answer had to be entered | disagreed | agreed |
| 13 | In my opinion, the layout of the items is clear. | disagreed | agreed |
| 14 | The hints and remarks after an answer has been checked are clear. | disagreed | agreed |
| 15 | In my opinion, the elaborations of the problems are clear. | disagreed | agreed |</p>
<table>
<thead>
<tr>
<th></th>
<th>Normally, with this kind of exercises, I prefer an extensive elaboration to a brief one.</th>
<th>disagreed</th>
<th>agreed</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>In my opinion, the screen construction is fast enough.</td>
<td>disagreed</td>
<td>agreed</td>
</tr>
</tbody>
</table>

Succession of items

<table>
<thead>
<tr>
<th></th>
<th>The choice of subject and level seemed logical to me.</th>
<th>disagreed</th>
<th>agreed</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>I want to be able to indicate myself which part of the course I want to practice.</td>
<td>disagreed</td>
<td>agreed</td>
</tr>
<tr>
<td>19</td>
<td>A system has been used that selects a next problem based on shown results. This principle appeals to me.</td>
<td>disagreed</td>
<td>agreed</td>
</tr>
</tbody>
</table>

My motives for my choice in question 20 are:

<table>
<thead>
<tr>
<th></th>
<th>It is handy that the system tunes the difficulty level to my performance.</th>
<th>disagreed</th>
<th>agreed</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>It is important to me to be able to decide for myself which problem to answer next.</td>
<td>disagreed</td>
<td>agreed</td>
</tr>
<tr>
<td>22</td>
<td>It is important to me to get different items with different sessions.</td>
<td>disagreed</td>
<td>agreed</td>
</tr>
</tbody>
</table>

Judgement

<table>
<thead>
<tr>
<th></th>
<th>My general judgement about this method of practicing is</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>With some modifications, the software could be adapted to be suitable for examination. My attitude towards this is</td>
<td>negative</td>
<td>positive</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remarks and suggestions for improvement:
## B.3 Questionnaire result data

This section contains the raw results of the questionnaire correlation analysis described in chapter 7. Table B.1 shows the correlation matrix for the first section of the questionnaire plus question 20 and 24, compared to all questions. The first number in a cell is the correlation coefficient, the number between parentheses is the P-value. For P-values smaller than or equal to 0.01, the cell is shown in boldface.

<table>
<thead>
<tr>
<th>Quest.</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>20</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>.5437</td>
<td>.1885</td>
<td>.2027</td>
<td>.2553</td>
<td>-.0951</td>
<td>-.2553</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.5437</td>
<td>.1386</td>
<td>.0362</td>
<td>.0032</td>
<td>-.1495</td>
<td>-.1215</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.1885</td>
<td>.1386</td>
<td>.5897</td>
<td>.3945</td>
<td>.3248</td>
<td>.3053</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.2027</td>
<td>.0362</td>
<td>.5897</td>
<td>.8107</td>
<td>.5197</td>
<td>.6823</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>.2553</td>
<td>.0032</td>
<td>.3945</td>
<td>.8107</td>
<td>.3269</td>
<td>.5600</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>.1949</td>
<td>.3660</td>
<td>-.1641</td>
<td>-.0320</td>
<td>.1963</td>
<td>-.2932</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>.0671</td>
<td>-.0813</td>
<td>-.878</td>
<td>-.1522</td>
<td>-.975</td>
<td>-.1328</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>.1099</td>
<td>-.0023</td>
<td>.0023</td>
<td>-.0558</td>
<td>-.2051</td>
<td>.2307</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>.0950</td>
<td>-.1435</td>
<td>-.1766</td>
<td>-.1835</td>
<td>-.0379</td>
<td>-.0972</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>.0877</td>
<td>-.0580</td>
<td>-.1894</td>
<td>.0632</td>
<td>.1224</td>
<td>.1081</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>.1581</td>
<td>.3931</td>
<td>-.1819</td>
<td>-.0082</td>
<td>-.0671</td>
<td>-.0183</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>-.0090</td>
<td>-.0449</td>
<td>-.2030</td>
<td>-.0082</td>
<td>.2440</td>
<td>-.1003</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>.0315</td>
<td>.3569</td>
<td>-.1826</td>
<td>-.2910</td>
<td>-.0580</td>
<td>-.0651</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>.1644</td>
<td>-.2901</td>
<td>.1209</td>
<td>.0638</td>
<td>.1855</td>
<td>.0931</td>
<td></td>
</tr>
</tbody>
</table>

### Table B.1. Correlation matrix of questionnaire result

*Note: P-values smaller than or equal to 0.01 are shown in boldface.*
Table B.1. Correlation matrix of questionnaire result (continued)

<table>
<thead>
<tr>
<th>Quest.</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>20</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>-.1046 (.302)</td>
<td>.0716 (.361)</td>
<td>-.0453 (.411)</td>
<td>0.0078 (.485)</td>
<td>-.0568 (.389)</td>
<td>-.1072 (.297)</td>
<td>-.0621 (.379)</td>
</tr>
<tr>
<td>17</td>
<td>-.1051 (.301)</td>
<td>-.2067 (.146)</td>
<td>-.1894 (.167)</td>
<td>.0395 (.421)</td>
<td>.1293 (.256)</td>
<td>.0575 (.386)</td>
<td>.1764 (.185)</td>
</tr>
<tr>
<td>18</td>
<td>.3127 (.056)</td>
<td>.0316 (.437)</td>
<td>-.0458 (.409)</td>
<td>-.304 (.439)</td>
<td>.1327 (.250)</td>
<td>-.0958 (.314)</td>
<td>-.0721 (.358)</td>
</tr>
<tr>
<td>19</td>
<td>-.1595 (.213)</td>
<td>-.0709 (.360)</td>
<td>.1399 (.239)</td>
<td>-.1512 (.221)</td>
<td>-.1069 (.294)</td>
<td>-.3212 (.048)</td>
<td>-.2437 (.106)</td>
</tr>
<tr>
<td>20</td>
<td>-.0951 (.319)</td>
<td>-.1495 (.224)</td>
<td>.3248 (.046)</td>
<td><strong>.5197 (.002)</strong></td>
<td>.3269 (.045)</td>
<td><strong>.4940 (.004)</strong></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>.3824 (.025)</td>
<td>.0586 (.384)</td>
<td><strong>.4383 (.010)</strong></td>
<td>.2954 (.063)</td>
<td>.3011 (.060)</td>
<td><strong>.5152 (.003)</strong></td>
<td>.0602 (.380)</td>
</tr>
<tr>
<td>22</td>
<td>.1501 (.227)</td>
<td>.3446 (.036)</td>
<td>-.0385 (.423)</td>
<td>.0376 (.425)</td>
<td>.0858 (.332)</td>
<td>-.3689 (.027)</td>
<td>-.1353 (.246)</td>
</tr>
<tr>
<td>23</td>
<td>.3580 (.033)</td>
<td>.0555 (.390)</td>
<td>-.1377 (.242)</td>
<td>-.2038 (.149)</td>
<td>-.2562 (.094)</td>
<td>.0825 (.338)</td>
<td>-.0973 (.311)</td>
</tr>
<tr>
<td>24</td>
<td>-.2553 (.099)</td>
<td>-.1215 (.269)</td>
<td>.3053 (.057)</td>
<td><strong>.6823 (.000)</strong></td>
<td><strong>.5600 (.001)</strong></td>
<td><strong>.4940 (.004)</strong></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>.0388 (.424)</td>
<td>-.0403 (.419)</td>
<td>-.0665 (.368)</td>
<td>.1994 (.155)</td>
<td>-.0243 (.451)</td>
<td>-.0369 (.426)</td>
<td>.3362 (.040)</td>
</tr>
</tbody>
</table>
The questionnaire
Appendix C

Using the World Wide Web for training applications

C.1 Introduction

During 1994, many people with access to the Internet discovered the World Wide Web (WWW, also called the Web). The Web provides a distributed hypermedia system; this is a powerful information delivery medium, which can be used very well for educational purposes. Started by CERN (the European Laboratory for Particle Physics) in Switzerland, the organisation is now a cooperation between CERN and the laboratory of computer science at MIT (the Massachusetts Institute of Technology). Also NCSA (the National Center for Supercomputer Applications) has been very active on this field. The WWW project uses hypertext and multimedia techniques to make the Web easy for anyone to browse, and contribute to.

C.2 The principle of WWW

Characteristic of hypertext is the pointer structure: a displayed text can contain specially indicated parts called anchors. When activated (usually clicked upon) they cause a new section of text to appear. For example, consider part of a text about means of transport as shown in figure C.1. If the user clicks the underlined word “cars”, a new page is shown explaining more about cars. That page in turn could incorporate pointers to subsystems of cars, such as the engine, the wheel, or to the construction process, the car industry, different types of cars, etc.

Hypermedia form a superset of hypertext: instead of just a text, a picture can be displayed, a sound can be produced, or even a movie or animation can be shown.
Means of transport

There are means of transport in many variants. Let us start with those that are in use at the moment. Some vehicles are designed to move on land, such as cars, bicycles and buses. Also on the land, but on rails, we find trains. If we look at the seas and rivers, we can see boats. Usually they are floating at the surface, but an exception is the submarine, which can be completely below the surface. We can also use the sky for transportation: think about aeroplanes and helicopters.

You might also want to take a look at the more old fashioned transportations.

---

Figure C.1. Example of a hypertext page

Strictly speaking, the hypertext example of figure C.1 uses hypermedia constructs: the buttons at the bottom are images.

In normal hypertext systems, the texts use internal pointers, meaning that only material which is available on the machine can be used. With the WWW, documents on other machines can also be accessed, via the Internet. The concept is like referencing a document, but instead of just a reference the complete document is available instantly.

To access the Web, a browser program is used. Usually, browsers have facilities to display pictures, and use external programs for delivery of sounds and animations. The browser reads documents, and can retrieve documents from other sources. To provide information on the Web, you can set up a server. Browsers can then access that server to retrieve and interpret documents.

C.3 User input with WWW

The construct as described until now is very suitable for distributed databases, or more generally, for presenting information. The Web will only be applicable for training purposes if there are more possibilities of handling user input, since answers must in some way be transported from the user to the system. There are facilities for these activities as well. Radio buttons can be used, texts can be entered, and buttons can be clicked. In response to such events, the server executes scripts in which the reaction to the event is specified. These scrips can execute external pro-
grams, acting on the input. These facilities are not very widely used yet, except for search engines, "white pages" (telephone and e-mail directories), and mail ordering companies, for example.

C.4 Discussion of usability

A very clear advantage of using the Web for delivery of educational material is the platform independence. Any platform for which a server is available, can be used to build the material to be delivered, and any machine for which a browser is available, can be used to receive the material. With the application of giving training in mind, another great advantage is the possibility for central storage of information while the students are able to work at home (via a modem), or in a terminal room at the department, without needing software other than a browser and possibly a sound or animation displayer. This browser software is readily available for all contemporary platforms, free of charge. Further, it is possible to provide additional information when desired, such as a list of formulae, and the students can give feedback on the system by e-mail. Finally, as the students are not able to reach the actual training program or the data files, system safety becomes a less important issue. The Web servers provide safety measures. These characteristics make a system based on WWW also suitable for training purposes in distance education.

A drawback of using the Web is that it is dependent on the network capacity. The load of the Internet has been increasing enormously the last year, mainly due to WWW activities and the growing interest of the non-university community. The likeliness of network overload in the future, and of problems due to network overload (especially long response times) will have to be examined.

C.5 Needed system modifications for WWW usage

Modifying the system to allow the use of the Web would mean a slightly different approach; at the moment the management system is the initiator and calls the items. With a Web based system, items would be displayed by the browser, and the server would be the initiating program. So, when an answer to an item is given (or when a session is started by entering an identification) the server would activate the management system. This program would read the necessary data, update the model, generate new pages containing an item and the current statistics, save the new data, and finish. Figure C.2 illustrates the principle. The server would provide the new page to the user who can answer it, causing the sequence to be restarted. To realise this, more data would have to be saved, such as the items that have already been
used this session. Further, time stamps are needed, to be able to make a decision if the student is in the same session or not. Because the management system is not always active during a session (i.e. a period in which the student is practising without long interrupts) but is restarted with every item answer, a time stamp is necessary to be able to see if a previous item was answered just before the latest one, which means in the same session, or longer before, for example on a previous day.

A possible cause of problems is the user’s freedom. It is not possible to control the actions of the students using a browser fully, therefore measures must be taken to prevent the students from “browsing back” to the previous page, giving another answer and continuing. Further, the accessibility must be taken into account. It is not always desired that people from outside the university, or even people not following the course, can access the system.

Considering the item delivery and the toolboxes as made for Taiga, the majority of functions can be translated quite easily. The picture standard in the Web is GIF, which is also the standard in the Toolbox for Taiga. The other graphical facilities would have to be translated to GIF as well. This task is quite straightforward; the only addition would be the encoding of the image into the GIF standard. Further, the interface with items would become different, since Taiga would no longer be used.

A drawback would be that existing items would have to be translated into a specific format called HTML (HyperText Mark-up Language). HTML does not include methods for formulae, super- and subscripts yet, but the language is still in development. Another drawback of using the Web would be that it would limit the choice of item implementation environments. Naturally it would be possible to automate the process of creating an HTML-page considerably, for example by using an HTML editor, so that the teacher can concentrate on item layout and contents.
Adaptieve instructie per computer in technisch-wetenschappelijk onderwijs

Oefenen wordt gezien als een belangrijk onderdeel in het leren van technisch-wetenschappelijke kennis. Om oefening effectief te laten zijn, moeten de studenten actief deelnemen. Teneinde dit te bereiken zou de groepsgrootte bij oefensessies klein moeten zijn opdat de docent snel en gericht terugkoppeling kan geven aan de student en hem of haar de oefeningen kan verschaffen die goed aansluiten bij het bereikte beheersingsniveau. Echter, deze situatie is onhaalbaar door de veelal beperkte personele omvang van de universitaire staf. Een oefensysteem per computer kan een oplossing bieden voor dit dilemma. Voor maximale produktiviteit moet het systeem de training aanpassen aan het niveau van de student. Het doel van het onderzoek is het realiseren van een adaptief oefensysteem voor gebruik in het technisch-wetenschappelijk onderwijs.

Er zijn al programma's voor computer-ondersteund onderwijs sinds het begin van de jaren zestig. Hoewel ze vaak inventief gemaakt zijn, was het merendeel van die vroege systemen vrij star en zeer afhankelijk van de anticipatie van de ontwerper. Sinds eind jaren zeventig zijn er andere mogelijkheden onderzocht; dit resulteerde onder meer in onderzoek op het gebied van intelligent tutoring systems (ITS) en itembanking. Dit laatste wordt normaliter gebruikt voor toetsing: de docent beschrijft een deelverzameling van een vraagstukkenbestand en laat het systeem verschillende toetsen genereren. Hoewel itembanking ook gebruikt kan worden voor oefenen, is een adaptieve methode te prefereren. Hiermee wordt een methode bedoeld die bij het selecteren van een vraagstuk rekening houdt met de prestaties van de student op voorgaande vraagstukken. Omdat technisch-wetenschappelijke
vakken vaak multidimensionaal (niet meetbaar in één onderliggende vaardigheid) zijn, wordt een adaptatiemechanisme in twee niveaus voorgesteld: een adaptatie op vakniveau, die een onderwerp selecteert, en een adaptatie op onderwerpniveau, die een item over het gekozen onderwerp selecteert. Student-modellen en vakmodellen, zoals gebruikt bij ITSen, zijn ook bruikbaar in een adaptief oefensysteem. Voorgesteld wordt om deze toe te passen, maar de domeinafhankelijke kennis in de vraagstukken te laten; dit verschaft een meer algemene bruikbaarheid.

Met het oog op mogelijke toepassing in de te ontwikkelen programmatuur zijn diverse methoden in de kunstmatige intelligentie onderzocht. Technieken voor kunstmatig leren lijken niet erg geschikt voor het voorgestelde trainingssysteem. Wel toepasbaar is een zogenaamde blackboard architectuur, waarbij onafhankelijke kennis eenheden een database delen.

De ontwikkelde programmatuur voor een algemene adaptieve sessiebeheerder bestaat uit een shell waarbij verschillende adaptatiemechanismen kunnen worden toegepast. Het systeem is gebaseerd op de twee niveaus zoals eerder beschreven, waarbij de uiteindelijke implementatie van de adaptatiemethoden niet gespecificeerd is. Vakken waarvoor de shell toepasbaar is, moeten gerepresenteerd kunnen worden in een taxonomie van onderwerpen of concepten die separaat gecategoriseerd kunnen worden. Er is een object-georiënteerde methode gebruikt, vooral ter ondersteuning van inkapseling. Het systeem is geïmplementeerd in C++ met het oog op overdraagbaarheid.

Bij de testvakken bestaat het model op vakniveau uit een en-graaf met voorkennisrelaties; het model op onderwerpniveau is gebaseerd op de Item Response Theory (IRT). Het selectiemechanisme dat in IRT gebruikt wordt, veronderstelt dat het niveau van de student constant blijft tijdens de sessie en kan resulteren in ongewenste grote verschillen in moeilijkheidsgraad tussen twee opeenvolgende vraagstukken. Daarom is een heuristisch aanpassingsmechanisme toegepast dat een geleidelijke variatie biedt en niet veronderstelt dat het niveau van de student constant blijft tijdens de sessie.

Het systeem is toegepast bij inleidende vakken Regeltechniek en Elektronica. Voor implementatie van vraagstukken werd vooral gebruik gemaakt van het auteursysteem Taiga. Om Taiga geschikt te maken is het uitgebreid met een interface naar het beheersysteem en een toolbox voor tonen van regeltechnische figuren en externe figuren en teksten.

Een test met 29 studenten is uitgevoerd bij het vak Elektronica. De docent heeft de vragen zelf geïmplementeerd in een vrij kort tijdsbestek, ondanks enkele beperkingen van Taiga voor een dergelijke toepassing. De steekproef studenten was niet groot genoeg om duidelijke conclusies te kunnen trekken, maar de docent oordeelde

Het hoofddoel van het onderzoek is bereikt. Een algemeen toepasbaar oefensysteem is nu beschikbaar. Echter, er moet nog het een en ander worden aangepast om het systeem efficiënt te kunnen gebruiken in de praktijk. Ter evaluatie van de prestaties van het systeem is het sterk aan te bevelen om enkele meer grootschalige sessies te organiseren met een grotere vragenbank en grotere groepen studenten. Verder zijn vier aanbevelingen:

- Geef het adaptatiemechanisme een meer adviserend karakter, om de studenten meer invloed te geven.
- Ontwikkel een methode om, gebaseerd op sessieresultaten, de item-parameters (zoals de moeilijkheidsgraad) aan te passen.
- Ontwikkel een studieadvies-functie.
- Maak het systeem beschikbaar via het World Wide Web voor een betere bereikbaarheid met betrekking tot tijd, plaats en platform.
Adaptieve instructie per computer in technisch-wetenschappelijk onderwijs
Curriculum Vitae

Rik Essenius was born in Rotterdam on February 23, 1965. He graduated from the "Erasmiaans Gymnasium" in Rotterdam in 1983. Then he studied Computer Science at the Delft University of Technology, and graduated in 1990 at the Control Laboratory of the Department of Electrical Engineering. From 1986 to 1990 he was also a part-time software developer for L&M Adviseurs b.v., an insurance consultancy in Rotterdam.

In October 1990 he started as a researcher at the Control Laboratory on a two-year project to implement a computer based training system for the course "Systems and Control Engineering". After two years, the project was extended with another two and a half years, in order to design and implement a more generally applicable system. In this period there was close cooperation with the Electronics Laboratory. The project was finished with a successful test of the system, adopted to the introductory course on Electronics.
Index

A
abstraction 36
active student participation 3
adaptation process 74
adaptive computer-based training 5
adaptive mastery testing 14
adaptive program control 11, 49, 53, 81
adaptive strategies 14
adaptive system
   analysis model 57
      basic assumptions 49
adaptive testing 5, 14
adaptive training 14
analogy 36
and-or graph 23
answer model 64
appropriately difficult items 21
AQ algorithm 37
artificial neural nets 43
authoring languages 10
authoring system 13, 96, 98

B
backward chaining 33, 68
bandwidth problem 4, 27
Bayesian posterior beta distribution 16
blackboard architecture 46, 68
Bode diagram 99, 106, 125

C
case based reasoning 40
CATE 8
command line parameter 85
computer-assisted instruction 9
   control strategies 11
   teaching strategies 10
   computer-based testing 12
   computer-based training 1, 4
   concept lattice method 23
   conceptual clustering 39
   concretion 36
   confidence interval 77
   control communication 83
   control object 68
   control scheme 53, 81
   control strategies
      adaptive program control 11, 49, 53, 81
      program control 11
      student advisory control 12, 29, 49, 53
      student control 11
   Control Toolbox for Taiga 98, 104, 117, 139
   correlation analysis 149, 152
   course description file 85
   course model 52, 58, 59, 61
   course selection 82
   course structure 74
   current directory 84
   curricular model 49

D
data file creator 86
data file mover 87
data mining 39
dataflow diagram 109
deduction 36
development system 13
dissimilation 36
distribution system 13
domain knowledge 50
drill and practice 10
Index

E
educational productivity 1
Electrical Engineering 6, 7
Electronics 89, 141
engineering education 1, 6
error variance 20
estimation error 76
executing external programs 87
expert systems 31
dependent 34
first generation 32
first generation limitations 33
second generation 34
external programs 86

F
feedback 54
file manager 57, 58, 65
forward chaining 32
frequency response 99, 123

G
General Toolbox for Taiga 97, 130, 139
generalisation 36
genetic algorithms 44
gradual increase in difficulty 26, 74, 77
guessing parameter 19, 23

H
heuristic adaptation 79
heuristic model updating 77
HTML 204
hypermedia 201
hypertext 201

I
individual training 5
induction 36
inference mechanism 69
inference patterns 36
inference types 35
influencer 64
influencing subject 80
information function 20, 76
Inigo 136
initialisation file 85
inquiry 10
intelligence 14
intelligent tutoring systems 10, 27
overlap in knowledge 28
process control metaphor 29
item 12, 98
item banking 12
item characteristic curve 17
item difficulty 17
item discrimination 17
item environments 13, 92
general programming language 96
Inigo 136
on-line simulation 136
requirements 92
Taiga 93
Taiga extensions 96
item form adaptation 25
item interface 54, 86
item information file 55, 186
item result file 55, 189
principle 55
item manager 57, 58, 62
item model 53, 64
item pool score 21
item presentation 82
Item Response Theory 17
Bayesian posterior estimation 21
drawbacks 75
extension for multiple attempts 22
proficiency 17
item selection 74, 82
item selection strategy 59, 81
Items-D 12, 50, 88, 98, 138
development system 13
distribution system 13
limitation 14
profile 13
session definition 13
structure 12
variation of parameters 13
Items-D type Interface for Taiga 97, 101, 109, 139

K
knowledge source 47, 68, 81
knowledge unit 23

L
learning by doing 3
learning by observation 38
learning from examples 37
likelihood function 19, 22
list class 71
M
machine learning 34
management system 99
Mason-type flow diagram 99, 107, 127, 135
mastery 15
mastery level 17
maximum likelihood 19
message dispatcher 67
moving average 78
multiple choice item 73
multiple classification item 73

N
Nyquist diagram 99, 106, 125, 135

O
object oriented design 51
objective of the product 7
one-parameter logistic model 17

P
platform independence 56, 71
pole-zero configuration 97, 99, 105, 107, 121, 137
prerequisite information 60
prerequisite knowledge 7
prerequisite subject 50, 73, 80
prior distribution 76
prior information 21
prior knowledge 14
probability ratio 15
proficiency 17
program control 11

Q
questionnaire 145, 193

R
raster image 130
raw score 21
reinforcement 2
reinforcement learning 41
resource directory 84
result directory 84
result extractor 86
run script 87

S
score assignment 21
self-tuning controller 71
Sequential Probability Ratio Test 15
series identification 53
session logger 57
session manager 73
similisation 36
simulation 4, 10, 136
specialisation 36
statistical analysis 147
stopping criterion 77
structure chart 109
student advisory control 12, 29, 49, 53
student control 11
student identification 67
student judgements 158
student model 52
student model maintenance 83
study advice 160
subject 60
subject lattice 74
subject model 59, 60
subject selection 82
supervised learning 43
swapper 88, 138
System and Control Engineering 89
system performance 158

T
Taiga 93, 160
taxonomy 49, 51, 52
teacher dependency 14
test set-up 141
tests 10
three-parameter logistic model 19
time response 99, 104, 118, 134
training 3
TRIP 138
t-test 150, 154
tutorials 11
two-parameter logistic model 17

U
user interface 56, 58, 66, 83

W
World Wide Web 170, 201