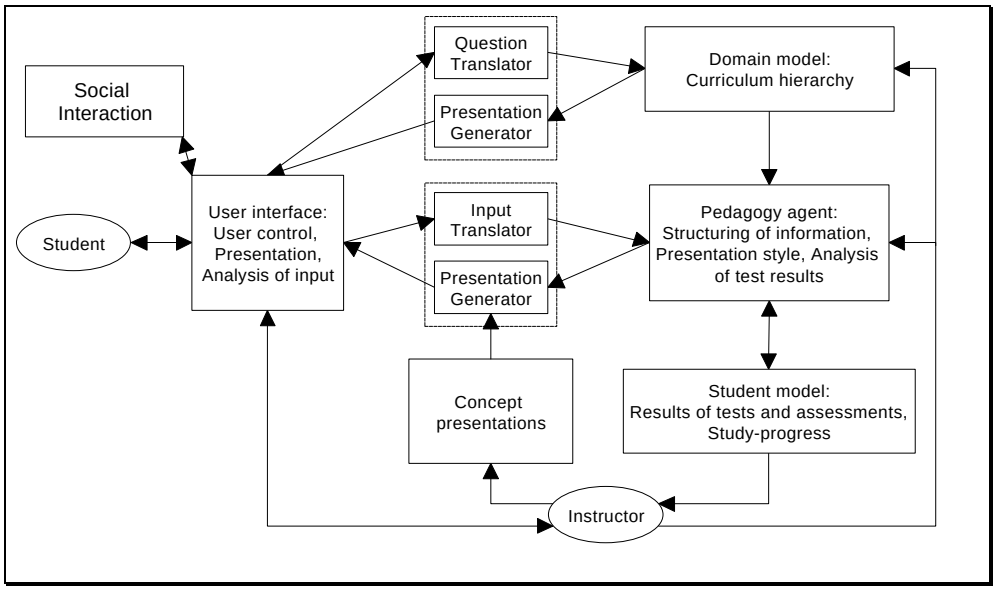
Prototyping an HPCNet distance learning environment: rationale and implementation

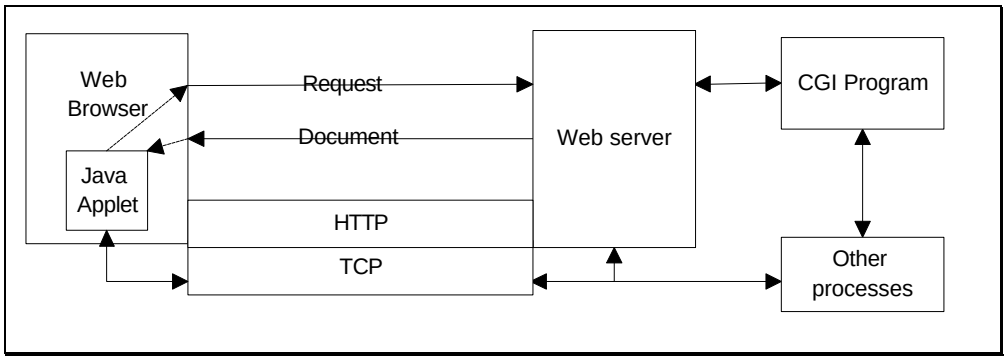
Braan, J.M.; Hoekstra, A.G.; Sloot, P.M.A.

Publication date
1998

University of Amsterdam, faculty of Computer Science

Prototyping a Distance Learning environment on the Internet: Rationale and Implementation





Master thesis by:
J.M. Braan

Supervisors:
Dr Ir A. Hoekstra
Prof. Dr P. Sloot

Voorwoord

De keuze voor het onderwerp ‘Distance Learning’ hangt nauw samen met mijn persoonlijke interesse op dit gebied. In mijn toenmalige bijbaan op Universiteit Nijenrode kwam ik al in aanraking met de materie, en rond de tijd dat ik op de Universiteit van Amsterdam begon met afstuderen startte Nijenrode een proefproject met onderwijs-ondersteuning via Internet. Het is dan ook niet verwonderlijk dat ik zo nu en dan de enkele uren die ik nog voor Nijenrode werkte aardige discussies voerde over dit onderwerp.
Hoewel dit van te voren al was verwacht, en daar ook aandacht aan was besteed, bleek het onderwerp een zeer inter-disciplinair karakter te hebben, waarvan de raakvlakken met mijn studie niet altijd voor de hand lagen. Met name de toch noodzakelijke keuze voor een bepaalde pedagogische richting bracht moeilijkheden met zich mee, omdat de experts op dit gebied elkaar nogal eens in de haren willen vliegen over de juiste keuze. Dit geldt ook voor mijn uiteindelijke keuze om zo dicht mogelijk bij een technisch georienteerde richting te blijven, maar voor iemand met een technisch georienteerde opleiding in plaats van een onderwijskundige is deze keuze niet vreemd. Het is zeker interessant om iets bijna van ‘scratch’ te moeten onderzoeken, maar het is soms lang zoeken geweest naar de juiste aanknopingspunten.
De bedrijfskundige kanten van mijn studie zijn vooral in het eerste hoofdstuk terug te vinden, maar vormen niet het hoofdonderdeel van deze scriptie. Wel zijn ze nuttig gebleken bij de keuze van een veranderingsrichting in het onderwijs met betrekking tot haalbaarheids- en risico-vraagstukken.
Hoewel geen volledig systeem is geimplementeerd, zijn er toch voldoende conclusies te trekken om een mogelijk volgende fase van systeem-ontwikkeling te kunnen starten. Ik zelf kan gelukkig mijn interesse in de vele kanten van ICT-ondersteuning voor onderwijs tot uiting brengen in mijn baan, zij het dan voor een andere universiteit.

Zoals het hoort bedank ik iedereen die me heeft geholpen bij het onderzoeken, (her-)schrijven en uiteindelijk het schrappen. Naast miin begeleiders zijn dit onder andere Robert Belleman, de heer Klaasse, de heer Groenenboom, de heer Kroodsma, Maurice van den Akker en natuurlijk mijn vriendin, haar familie en mijn eigen familie.

Jeroen Braan
Amsterdam, 2 april 1998

0. Introduction

0.1 Basic approach

Currently, the High Performance Computing Network of Excellence (HPCNet) is looking into the possibilities of using Distance Learning to supply students with courses on their subject domain in a time and place independent manner. They focus on using the Internet and more specifically the World Wide Web to accomplish this. However, no detailed description of what such an environment should look like exists, and therefore no prototype of such a system has been implemented. Basic requirements can be described as the need to provide a time and place independent course within an integrated environment. These requirements form the basis of this thesis, and it therefore focuses on building a prototype for such an environment almost from scratch. This approach has been translated into the basic question of how well the World Wide Web is fitted to function as a platform for an integrated Distance Learning environment. The goal of this research can therefore be described as:

> ***Providing HPCnet with insight into how to use the Internet, and more specifically the World Wide Web, to provide students with a time and place independent way of participating in courses on High Performance Computing.***

Since this is a rather broad goal, a specific case example is used, consisting of a two weeks summer course on Parallel- and Distributed Computing. The major question this thesis tries to answer is therefore:

> ***Which components should be present in an integrated Distance Learning system for the specific summer course, and how can these components be implemented using the World Wide Web as a platform?***

For this, I have approached the problem at a standard way used for building information systems: combining demands from the business domain with the possibilities of the technology involved. My approach towards this method involves a number of steps. First, the introduction of ICT within the educational domain is reviewed at a broad level, resulting in an approach towards developing the prototype. Based on this approach, three steps can be identified: requirements planning, risk analysis and the creation of an initial software prototype. A more detailed description can be found in the first chapter.

0.2 Definition of 'integrated Distance Learning system'

First of all, the term 'integrated Distance Learning system' should be defined in more detail. Based on the description of 'tele-learning' presented in (Collis, 1997), the definition for the above mentioned term is constructed. According to Collis, tele-learning is: *making connections among persons and resources through communication technologies for learning-related purposes*. For this thesis, that definition is to broad. Therefore, I have changed it into: *an integrated Distance Learning systems provides an integrated educational environment in which students can interact with domain specific contents, peers and instructors in a time and place independent manner to gain knowledge about that domain.* Integrated in this respect means that the different components of such an environment are presented to the student through the use of a single interface, and are connected within a single logical system aimed at

educating the student. Time and place independence in this definition is based on three factors, found in literature. The first of these aspects is what can be described by:

- Any student should be able to follow a course at a different time and location than any other student;

This is a definition that is very common in articles about distance learning, and focuses on the use of networks. However, the examples of time and place independent studying are also talking about the freedom of a student to do work for a course, interrupt this work and continue at any other time (or place) (see for example (Ives, 1996) or (Kalakota, 1996)). So another aspect of time and place independence can be described as follows:

- Any student should be able to interrupt a course at any moment, and continue at any other time and/or location;

Thus creating the need for a distance learning environment to keep track of a student's progress. Next to this, in (Kalakota, 1996) Kalakota and Whinston mention one more aspect of time independence:

- Any student should be able to follow a course at his or her own pace

The organization of a course should provide support for this aspect, creating problems when courses need to be updated.

0.3 Sub-questions

The demands and expectations put on such a system can be divided into general and specific demands and expectations. General demands follow from applying the results of a study into using ICT within an educational environment to the specific situations. Specific demands can at first instance be described as the need for the system to adapt itself to the knowledge level of a student, and providing students with the possibility to experiment on a real parallel machine. The creation of a prototype for such systems requires a large amount of time and resources. Therefore, this thesis focuses on a subset of the requirements. Next to testing a student's knowledge level, the creation of a practical assignment environment was of major importance. The use of multimedia, a tool guiding a student through the environment and the integration of social interaction in the system are described, but not in great detail.

The approach chosen is focused on answering the following sub-questions:

- How can information- and communication technology help in changing the educational process?
- What kind of functionality should the required software offer?
- Which components should be present when the system is build using client/server based information retrieval?
- How should the server components be build?
- Can the World Wide Web theoretically support such an environment?
- What kind of functionality do existing World Wide Web based systems offer?
- How can a Virtual Computer Lab be implemented?

Resulting in the writing of the following seven chapters:

Educational Context	*Chapter 1*: Changing the educational process using ICT
Effects on software development	*Chapter 2*: Influence of pedagogical choices on software requirements
Modeling the environment	*Chapter 3*: Placement of different components within an integrated framework
Requirements of server components	*Chapter 4*: Use of the conceptual structure of a domain for student modeling and expert help
Limitations of using the Internet and the World Wide Web	*Chapter 5*: Technical description of the Internet and the World Wide Web
Current Distance Learning environments on the Internet	*Chapter 6*: Description of Distance Learning environments currently available on the Internet
Including a practical assignment environment	*Chapter 7*: Description and discussion of an implementation of a Virtual Computer Lab prototype

1. Introducing ICT in the educational process

1.1 Global view of education

In general, educational institutions should be focused on at least three goals: ensuring retention, understanding and active use of knowledge and skills (Perkins, 1991). Based on these goals, (Somekh, 1996) identifies five different levels of learning, which can be summarized as follows:

A. This type of learning focuses on recognition. It requires matching of superficial features of information presented to information previously presented.
B. The main focus here is on recall. The student is superficially engaged with the content and is reproductive.
C. The focus here is on reconstructive understanding or comprehension of statements, concepts or principles. It involves semantic interaction with content and the student is reconstructive.
D. The student here is globally reconstructive or has an 'intuitive' understanding. Learning is experiential and aimed at problem-solving. The student should master the discipline, and the focus is on the structure of that discipline.
E. In the last type of learning, the focus is on constructive understanding. The student 'creates' fields of knowledge through exploration and problem-finding. This type is domain-dependent, and discipline-independent.

The current way institutions which focus on formal higher education try to reach these goals is largely based on time and place dependence. Students have to come to a university, where an instructor uses subject matter and instruction to introduce a student in the instructor's area of expertise, and teaches the student how to move around in this area (Kirschner, 1995). The organization of institutions providing such a form of education can be globally divided into three layers, as depicted in Figure 1-1 (Kirschner, 1995). Current developments focus on new ways to reach the mentioned goals, supported by information- and communication technology (ICT) in all of the three layers. This thesis focuses on using ICT in the primary educational process only.

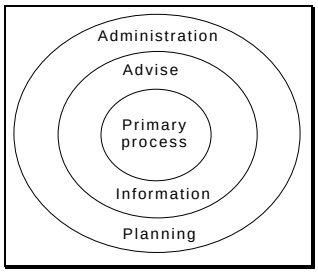


Figure 1-1 - Layers of educational processes, (Kirschner, 1995)

1.2 Components of the primary process

According to Collis, the pedagogical components found in the primary process, can be described by a so called 'pedagogical profile' (Collis, 1996). This profile is based on the notion that any course consists of some combination of one or more of the seven components described below (from a student's perspective).

1. Attend lecture (with differing amounts of student questioning and student interaction);
2. Personal contact with instructor;
3. Discussion among students;
4. Self-study, reading;
5. Do an assignment, individually (usually with feedback from the instructor);
6. Do an assignment, as a group (usually with feedback from the instructor);
7. Take an examination (to make evaluation of student performance possible).

In general, a weighted combination of these components can be used to define a course. Information and Communication Technology can provide benefits in each of these components, and can either change (enrich) components in such a way that the profile stays the same, or change this profile by changing the distribution of student time (pedagogical re-engineering).

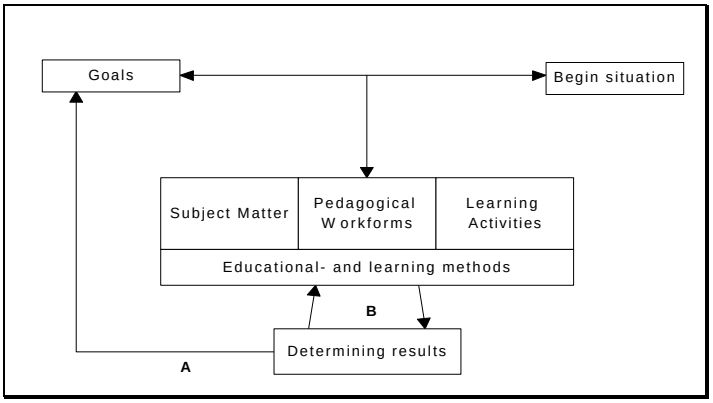


Figure 1-2 - Components in the educational process, (Dousma, 1995)

The way current educational institutions have implemented the primary process can be described by the components found in (Dousma, 1995), and is depicted in Figure 1-2. The goals in this figure present the ideas of the instructor and student about what they want to reach with education. The instructor has to take into account the knowledge present in the group of students, and the students have to ask themselves whether their knowledge is sufficient for the starting situation. The instructor decides which subject matter will be used, and why. The pedagogical components are chosen, and the student activities are defined. The instructor also decides on what educational resources to use to reach the defined goals. The last component is the component in which the results are identified. The arrow labeled 'A', represents the most important function of testing in such an environment: examining whether the goals are reached (product evaluation). The arrows labeled 'B', represent the other function of testing: examining how the educational process can be improved (process evaluation).

1.3 Ideas about the new educational environment

Much literature focuses on combining modern information- and communication technology (ICT) with new pedagogical insights to provide solutions to a number of problems with the current educational model. Basically, four major problem-areas can be identified. The first of these areas can be summarized by findings of a research described in (Duguet, 1995). Here, it is stated that economic and social changes are generating an increasing demand for higher and continuing education from secondary school-leavers, working adults, jobless and retired people and part-time workers. This results in three problems: time available to working adults is limited and does generally not fit in with conventional university hours, the diversity of people's learning requirements makes teaching supply more specialized and people working or living a long way from institutions that interest them, or physically handicapped people, have many problems with the location of teaching.
The second problem-area focuses on the increasing costs of traditional education, due to the fact that employers and individuals who benefit from an investment in learning are expected increasingly to contribute (Duguet, 1995). This makes it necessary to reduce the principal costs of traditional education: the wage bill and the construction and maintenance of buildings (Minoli, 1996).
As a third problem-area, rapidly aging knowledge (and therefore teaching contents) in a number of disciplines is named. For students this means that lifelong 'just-in-time' learning will become necessary. This can be done by twenty-four hour on-line education and the change of providing a discrete, career-spanning set of concepts and tools into providing the skills and motivation for lifelong learning (see for instance (Ives, 1996), (Hämäläinen, 1996) or (Kalakota, 1996)).
Finally, it is said that people who have graduated are presented with an 'information explosion' and should thus be able to find the right information, and know how to handle it (Pilot, 1994).

The current educational model however, is still too much based on presenting-, absorbing- and reproducing knowledge.

Modern approaches towards providing education should therefore focus on time and place independence, individualized and up-to-date learning material and new teaching methods while decreasing the costs of supporting a growing student population. Providing ICT-based distance learning to enable an instructor to support a larger group of students without introducing the need for new buildings is seen as a viable solution.

1.4 Changes in pedagogical insights

An often used approach to describe the existing learning theories is by placing them somewhere between two extremes called 'objectvism' and 'constructivism' (Jonassen, 1991). Objectivists believe in the existence of reliable knowledge about the world. They assume that the world is structured and that the structure can be modeled for the learner. Learning therefore consists of assimilating the objective reality. On the other hand, constructivism claims that the reality is more in the mind of the 'knower' and that this knower constructs or interprets a reality based on his or her experiences (Jonassen, 1991). The most extreme form of constructivism focuses completely on the learner as a constructor of knowledge, so designing instruction is unacceptable (Wasson, 1996). As said, the different ideas about education are generally placed on the above described continuum, as is for example done in (Jonassen, 1991):

Programmed Instruction is based on the behaviourist ideas of Skinner who sees learning as conditioning students to produce desired responses to a carefully planned sequence of stimuli (Somekh, 1996), (Schank, 1994). The goal of learning was for the student to exhibit appropriate behaviours (Schank, 1994). Although these theories are outdated, they have been used extensively in the past to create educational software. A modern approach which is sometimes referred to as 'neo-behavioral' is called 'instructivism', and is partly based on Instructional Design (ID) methods. Instructivists intertwine behavioral philosophies and modern cognitive theories of instructional design, and focus on content selection, sequencing, structuring and presentation (Wasson, 1996). Currently, most of the discussion is between those in favor of instructional design, and those favoring a constructivist approach towards learning.

1.5 Introduction of ICT in the educational process

In literature, introducing ICT in the curriculum is described in a number of ways, primarily focusing on two points of view: changes in the primary process and changes in the educational environment. Combining these two aspects, five phases can be globally identified (based on (Kirschner, 1995), (Collis, 1996), (Mirande, 1995), (Geoghegan, 1994) and (Berghout, 1995)):

- Substitution, where ICT is used to increase the efficiency of education. Introduction of ICT occurs primarily at the second and third layer. Changes in the primary educational process can be hardly identified. The effects noticeable for instructors are primarily in using computers for administrative tasks;

- Enrichment, where ICT is used to change the components of the primary process without creating the need for changes in the second and third layer. One or more instructors have knowledge about using Computer Aided Instruction (CAI), and those instructors use CAI within the settings of their own lessons. The need for a policy focused on CAI arises;
- Innovation, where the primary process changes in such a way that this requires changes in the educational environment as well. The situation is reflected, resulting in a systematic approach for the continuation of CAI projects. A research into the possibilities of CAI is conducted, choices are justified, development takes place by professionals and the introduction of CAI is based on a plan;
- Integration, where an infrastructure exists which is in use and which supports the further introduction of CAI. CAI has become an integral part of the means used to reach the goals the institution has set for itself;
- Transformation, where education itself changes in such a way that it can no longer be described in terms used in the previous paragraphs. Both the primary process as well as the educational environment are completely different from the current situation.

The history of distance education itself is generally described in three generations. Around the end of the 19th century it was based on correspondence teaching. Starting in the late 1960s, media delivery based on radio, television, telephone and more recently the personal computer allowed the integration of different media, resulting in a focus on multimedia teaching concepts. The last couple of years, the advance of modern Information and Communication Technology has resulted into a movement towards completely integrated electronic learning environments. For instance, the Open University in the Netherlands is moving towards the third generation of distance education (Gastkemper, 1997), which means a movement towards innovation and integration.

1.6 Creating a convincing example

The path from enrichment towards innovation has proven to be very difficult. Different researches describe similar problems, where the role of instructors is often seen as being of primary importance in the innovation process (Kirschner, 1995). One of the proposed paths to follow is what Geoghegan calls creating ‘a compelling application’ and what is called a ‘convincing example’ by Mirande. The basis for creating a convincing example can be seen as a combination of a prototype approach and formative evaluation methods guiding the design.

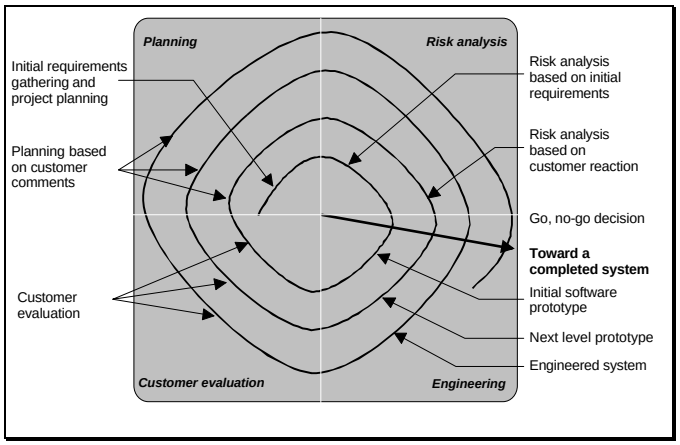


Figure 1-3 - The spiral model, (Pressman, 1992)

A process which can be used for creating such a prototype is described in (Mast, 1995), and is basically similar to software engineering methods such as described in (Pressman, 1992). This model is based on first creating a macro, meso and micro design, before realizing a technical design through prototyping. The macro design is concerned with the educational context, the meso design with pedagogical aspects and the micro design with all needed interactions within the system. The approach I have taken is based on the ‘Spiral Model’, which has been developed to encompass

the best features of both the classical life cycle and prototyping, while adding risk analysis (Pressman, 1992). This model is depicted in Figure 1-3.
This project must be placed at the beginning of the spiral, moving from 'Initial requirements gathering and project planning' to 'Initial software prototype'. It is quite common for a prototype in this phase to partly exist on paper, instead of being completely implemented (Pressman, 1992). Of the technical risks described by Pressman, I focus on risks having to do with implementation.
The steps I have followed can be described by:

1. Initial requirements gathering at a macro, meso and micro level. This step will be described in chapters two through four;
2. Determination of constraints and technical risk analysis: research into the use of Internet for such projects, based on confronting the results of step 1 with the (im-)possibilities Internet presents as well as a research into the functionalities offered by existing Distance Learning applications on the Internet. Chapters five and six describe this step;
3. Prototyping the needed system: both by describing and extending theories and applications from others as well as creating an environment in which a programming assignment can be done. Chapter seven describes the prototype.

2. Functionality in a mild constructivist system

2.1 Introduction

Some of the criticism of instructional developers on extreme constructivism can be found in (Dick, 1991). Basically, he states that extreme constructivism seems to try to cover too much, has little concern for entry behaviour of students and shows little apparent concern for certifying the competency level of individual students. Thus, with respect to the phases presented in chapter one, extreme constructivism can be seen as a transformation of the educational process, and is still in a somewhat experimental stage. The current focus therefore seems to be on what is called 'mild constructivism', and modern forms of instructional design are said to be consistent with such a moderate approach (Merrill, 1991). My choice for this approach towards constructivism is based on the following observations:

- extreme constructivism represents a leap towards transformation of the educational process, where risk of failure is generally high;
- constructivism seems to be lacking in any agreed upon definition which can be used for defining technical implications. Technical implementations of constructivist ideas seem to be experimental;
- although extreme constructivism is still a major area of discussion, more moderate approaches towards constructivism seem to be more agreed upon. Leaving pedagogical discussions to educational experts, the design of the system presented in this thesis therefore focuses on what seems to be the current state of agreement.

This chapter is thus based on modern instructional technology, which combines instructional planning and cognitive theories of learning. Instructional planning in this context is described as being 'the process of mapping out a global sequence of instructional goals and actions that enable the system to provide consistency, coherence, and continuity throughout an instructional session and enables this global sequence to be interspersed with local goals generated when instructional opportunities arise' (Wasson, 1996). Next to this, two areas of importance named in (Wasson, 1996) are also considered: the social nature of the learning environment and the amount of instructional decision making assigned to the computer system. It is not too difficult to create educational software ensuring learning at levels A and B of the topology presented in chapter 1. However, modern software should at least ensure level C learning, while trying to move towards the higher levels. Combining the five pedagogical design issues of educational software found in (Kanselaar, 1991) with aspects found in for instance (Somekh, 1996) and (Joyce, 1984), results in the following five area's of interest:

1. The structure of a course;
2. The interaction with the system;
3. Motivating students;
4. Providing feedback to a student;
5. Providing social interactions between a student and his or her peers or instructors.

Each of these aspects is examined in more detail in the following parts.

2.2 The structure of a course

The theory which forms the basis of constructivist learning is that each individual develops mental schema or 'mind maps' which serve to inform future thinking or action (Somekh,

1996). In (Joyce, 1984), the most important implication for teaching when focusing on mind maps, is the use of so called 'advance organizers'. These organizers enable students to have advanced knowledge of the ground to be covered.

2.2.1 Mind maps

According to (Somekh, 1996), Effective learning depends on the creation of new schema, or on existing schema being revised, extended or reconstructed. However, there is little understanding of how humans actually learn. The basic assumption is that 'learning results in the organizing of memory into structures' (Merril, 1991), which is assumed to be comparable to the way concepts are related within the conceptual structure of a discipline. Such a cognitive structure can be pictured as a hierarchical structured collection of ideas, in which new information and ideas can be stored, as long as this information somehow connects to the information already present in the structure. Figure 2-1 describes an example of such a structure. According to (Joyce, 1984), Ausubel states that new ideas need to be connected to the existing cognitive structure. If new ideas or information conflict too much with this structure, or cannot be connected to it, they will not be included in the structure or will not be remembered. Therefore the instructor should structure the presented knowledge in such a way that there are enough connection-points, so the students can connect the new information with their existing structure. This has resulted in course structures resembling Figure 2-1, used for intelligent reasoning by the system as described below. In chapter 4 and 5, these ideas are worked out in more detail.

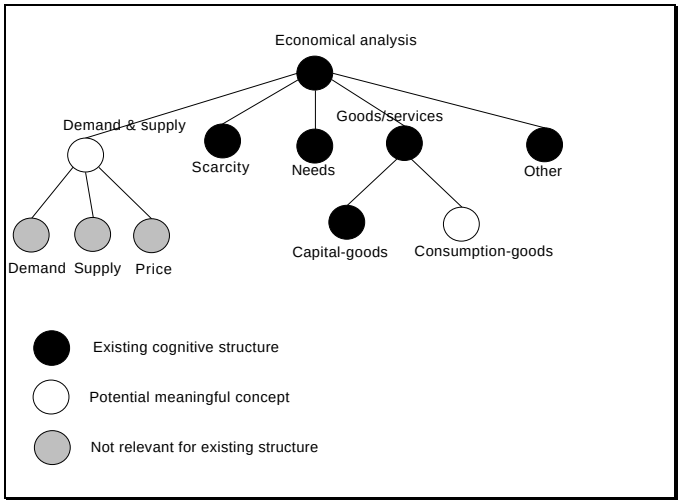


Figure 2-1 - Someone's cognitive structure, with respect to economy (Joyce, 1984)

According to Somekh, the mind map theory has two implications for software development:

1. To enable individuals to build on their existing schema, learners need to exercise control over their learning. Pacing and direction of learning are guided by questions arising from their developing understanding. Using user modeling is one of the ways this problem is addressed in educational software. Using pre-tests which will act to select appropriate material for a particular user may be the best way of attempting to address this problem;
2. In order to learn, individuals need support in constructing new schema. This suggests the need for structure underpinning the software design. Navigation is the centrally most important issue in developing software in a hypertext environment. Learners build their mental schema on 'where they go' and 'what they do'.

However, it is generally understood that a hypertext environment presents at least two major problems (Conklin, 1987). The first problem is called 'Getting lost in space', meaning that users need to know where they are and how they can get to some other place which is known to exist in the network. This problem is also known as the 'disorientation problem'. The next problem is called the 'cognitive task scheduling problem': the problems associated with the additional effort and concentration necessary to maintain several tasks or trails at one time.

Therefore, a conflict between providing user-control and pre-structuring information can exist. A structured environment should be presented to the student, but within the boundaries of this structure, the student should be able to control his or her environment.

2.2.2 Advance organizers

In (Joyce, 1984), the most important implication for teaching when focusing on mind maps, is the use of so called 'advance organizers'. These organizers enable students to have advance knowledge of the ground to be covered (Somekh, 1996). This can for example be in the form of introductory material or a set of learning objectives, offered before the actual learning activities take place and which is more abstract and general than the learning activities (Joyce, 1984). There is then much greater likelihood that the individual can develop meaningful schema which link with their existing schema (Somekh, 1996). Hypermedia or multimedia software can be used to present graphical overviews to serve as advance organizers, but merely providing such an advance organizer will not be sufficient (Somekh, 1996). Somekh says that students need to revisit concepts, until they are securely laid down in well developed mental schema capable of enabling intuitive decision-making. Introducing learners to a series of concepts, each one in a number of different ways, over a period of time, should be favored above repeating the same presentation of concepts over and over again. This brings with it the need of different representations of the same subject matter.
As another aid to developing schema Somekh names on-screen note-making facilities. Students can record their developing mental schema of the material, with the option of adding hand-written notes to the print-out at a later stage.

2.3 Interaction with the system

In (Wasson, 1996), the importance of authenticity of the learning task and the context in which the student works are identified. Much emphasis is being placed on the need for learning to be 'situated' to enable the construction of schema. One of the main problems in any learning is the need to translate concepts from a symbol system into meaningful schema.

2.3.1 Situated or authentic learning

The theory of authentic learning suggests that much learning is made more difficult because of the abstract context of learning (Somekh, 1996). Simulations can be used to create an authentic learning environment. However, Somekh states that simulations may necessitate the user to learn a complex set of rules governing the computer context of the simulation which are unrelated to the 'authentic' context. Therefore, when simulations or authentic environments are included within a distance learning environment, these have to be created in such a way that neither the rules explicit to such environments nor the presented solutions to problems should limit the learner in his or her understanding of the concepts. This can for instance be done by making sure the rules for simulation software are relevant within a real life situation.

2.3.2 Translation between symbol systems

In (Somekh, 1996), this problem is described in as follows: 'one of the most difficult problems facing software designers is how to get across sufficient content to enable learners to engage in problem-solving. A related problem, is how to get across sufficient information about a microworld or the rules of a simulation to enable learners to function intuitively with them.' She says that in general, the more interactive the software becomes, the more it is possible to

allow learners to learn by exploring models, or building models, rather than by digesting text. However, she acknowledges that designing systems that supports this kind of learning requires the use of complex software. She also notes that there are great differences between individual learners when it comes to the relative difficulty they experience in translating from different symbol systems, creating the need for an adaptive environment.

2.4 Motivating students

Motivating students is seen as being essential to learning. To do this, an educational system should provide an appealing environment, without interrupting the student too much. However, enabling a high level of concentration using merely a computer screen with a mouse and a keyboard is very unlikely.

2.4.1 Wanting

In (Richards, 1996) the ideas of Race are described, who uses the term ‘wanting’. According to Race, wanting implies much more than motivation since it lies at the center of ‘human urges’. Highly motivated individuals often produce excellent results. Richards identifies two basic ways for embedding wanting: increased value and increased enjoyment. Increased value is similar to the theory behind using advance organizers. The idea is that by showing students the purpose of studying each aspect of the course, and providing a schema of how the various aspects of the course link together, they get more motivated. Creating a stimulating environment, (called ‘micro-world’ by Somekh) increases the enjoyment of the student. By presenting the learning in a more enjoyable format, motivation can be increased. Using multimedia instead of plain text can be used to increase enjoyment. However, the media richness theory (Trevino, 1992) states that rich media can result in confusion and surplus meaning in cases where unequivocal messages should be delivered. Another study, presented in (Rice, 1993) concludes that multimedia systems don’t necessarily lead to better results, and can therefore be evaluated negatively by users. There is no standard way to decide on whether or not to use multimedia within an information system, but the media richness theory describes that an equivocal message might be better transmitted by a medium able to provide instant feedback, multiple cues, the use of natural language and a personal focus. Looking at the levels of learning presented in chapter one, it is clear that at the lowest level the concentration of unequivocal messages is the highest, thus reducing the profits gained since using multimedia could possibly result in communication failure. At the higher levels, using multimedia might result in better learning results through more effective communication.

2.4.2 Interruption of thought processes by the computer

Interactive software is unlikely to establish a high level of concentration if the learner’s thought processes are interrupted too frequently by the need to carry out tasks imposed by the computer (Somekh, 1996).
Richards warns for the problem of too much reliance on the relatively passive interaction involved in page turning (Richards, 1996). In such an environment, the user is not interacting with the information in any meaningful sense. In (Somekh, 1996) so called ‘flow’ is described. Flow is defined as an intense level of concentration where a learner may lose all sense of the context of learning, the pace of learning accelerates, and there is greatly enhanced motivation which spills over into a sense of excitement and continuing reflection after the event. However, since currently the standard ways of interaction for a user are presented by keyboard and mouse, achieving such a state is very hard. And, as can be read in for instance (Kirschner, 1995), the environment of a student is important for concentration. Basically, a computer screen is only a

small part of the total environment, and loss of concentration due to this environment is likely to occur. Solutions to this problem involve creating the right environment outside of the computer, which is not the subject of this thesis.

2.4.3 Real-time interaction

The notion of real-time with respect to both time dependent and time independent communication is an important subject. In a time dependent environment, real-time communication has to do with receiving a steady stream of information with low delay, making it possible to have two-way communication. In time-independent communication, the notion of real-time has to do with the time which passes between a user-action and a user-noticeable system-reaction. The choice which has to be made is whether to choose for a model based on downloading or a model based on streaming (such as television). A streaming model is the best choice when there is much focus on providing the user with the requested information as quickly as possible, since the user can get irritated when he or she has to wait too long for a file to be downloaded. In general, it is said that after 10 to 20 seconds of inactivity a user gets unhappy with the response of the system. In a networking environment, so called “jitter” (packets of information transmitted at exactly regular intervals will arrive at somewhat irregular intervals) has to be taken into account.

2.5 Feedback

Ideas about feedback focus on two distinguishable points: providing possibilities for self-reflection and allowing students to have a sense of their progression.

2.5.1 Meta-cognition

Meta-cognition is basically a term for critical self-reflection on oneself as a learner. In (Somekh, 1996) it is stated that there is a role for education in ‘enhancing the acquisition of meta-cognitive skills and learning strategies through explication of, and reflection on the learner’s knowledge as well as on their thinking methods and learning activities’. Pre-tests can include some element of analysis of an individual’s learning style which can be fed back to the user (Somekh, 1996). Another solution is to provide the already mentioned on-screen note-book for recording meta-cognitive reflections. As will be seen in chapter 5, automated pre-testing can be done by combining test theory and Bayesian networks at the lower knowledge levels. In general, assessment of student’s knowledge presents a bottleneck within the development of educational software as is described next.

2.5.2 Feedback and assessment

According to Richards, students must have some sense of their progression, in order to improve their skills or knowledge. He states that a problem encountered in many hypermedia learning products is that students are often unable to see how they are progressing with regard to the learning goals, and may therefore rapidly lose interest (Richards, 1996). Somekh says that the main problem of computer-mediated assessment is that it is difficult to prevent testing from emphasizing the lowest A and B levels of the typology presented in chapter 1, at the expense of even the C level. This is simply because the former are so much easier to test than the latter. Computer assessment of higher levels of learning demand considerable sophistication in software design and programming.
However, Somekh says that leaving out formal assessment might result in a situation where students view the software as an optional extra, and lecturers will not recommend the software strongly because they will have no way of checking whether or not students have used it.

Clearly, there is a need for both feedback and assessment, but it is not likely that computer support can be created for levels of learning exceeding level C. As will be described in chapter 4, it is at these higher levels that the availability of a tutor is most needed. Much of the work involved in assessment of lower levels of learning can however be done automatically, and it is at these levels that automatic pre-testing will be most successful.

2.6 Student-tutor and student-student interaction

In literature, the term 'zone of proximal development' is often encountered. Vygotsky says that learners have limits to their current level of achievement, beyond which they are unable to go without the interventions of a teacher or peer (Somekh, 1996). With the help of such interventions, learners can go beyond these limits into their 'zone of proximal development', and such experiences serve to extend their understanding so that next time they may be able to achieve this level of understanding without such mental and social 'scaffolding'.

2.6.1 Social interaction

Social interaction might provide a very effective way to provide mental scaffolding. As a result of the recognition of the social nature of learning, many writers place emphasis on the importance of discussion in facilitating learning (Somekh, 1996). A dialogue between instructor and student can be used to analyze the student's understanding, and function as a basis for subsequent dialogue (Brailsford, 1997). Thus creating the possibility to target feedback specifically at individuals, taking into consideration individual personalities, abilities and expectations (Richards, 1996). Peer discussions are not only useful to provide a social environment, but also create an environment in which 'new knowledge [is] acquired by a recursive process of debate which stems from previous knowledge. This newly acquired knowledge is then used as the basis for continuing debate.' (Brailsford, 1997). These aspects of interaction create the need for a more social environment within courseware. The preferred position within the different learning levels is described next.

2.6.2 Different types of courseware

The different types of courseware currently available can be described by a number of variables. One approach used in (Brailsford, 1997) and (Lotus, 1996) can be described by the following four paradigms:

1. primary courseware. Here, pieces of courseware are intended for the primary exposition of subject material, and tend to be declarative and instructor-led by nature;
2. secondary courseware. This kind of courseware is based on the mild constructivist model, allowing learners to work on their own, possibly isolating them;
3. tertiary courseware. This type of courseware combines secondary courseware with the communication and collaboration aspects of classroom-based instruction;
4. Learning team centered courseware. This type of courseware is only described in (Lotus, 1996), and is more focused on communication and collaboration than tertiary courseware. In this environment, knowledge emerges and is shared through collaboration of individuals within learning teams providing the right environment for courses aimed at mental model change.

The fourth paradigm differs from the previous in that the primary focus is on collaboration, and that the primary focus in the third paradigm is on the individual learning experience of a student, combined with collaborative aspects of classroom-based learning. According to Brailsford et al, a combination of instructor-led, student-centered and collaborative courseware

should be present in supporting the learning process, since this can be described by a perpetual cycle of three phases: conceptualization of subject material from declarative sources, construction (such as essay writing) and dialogue (Brailsford, 1997). In an environment based on individualism (such as a time- and place independent system), communication with instructors and peers can help learners to go into their 'zone of proximal development'.

2.7 Conclusions

Under the influence of different factors, a combination of mild constructivism and the use of ICT for education presents a powerful new paradigm for computer assisted learning.

For the system described in this thesis, the following consequences are of interest:

- more insight into the use of computer based testing to enable student modeling for adapting a course to a student's knowledge level is needed;
- an educational system should be based on a well-defined structure of the domain, which forms the basis for navigation and advance organizers as well as student modeling;
- rules in presenting a practical assignment environment should be relevant within a real life situation;
- interaction between a student and the system can become a bottleneck, interactive multimedia can be of help here;
- different representations of the same subject matter should be available;
- the use of annotation can help the student for recording meta-cognitive reflections and as an addition to advance organizers. This requires at least the technical capability of connecting sometimes large pieces of text with concepts. Such notes should be stored centrally due to the place independent nature of the system;
- interventions of a teacher or peer are needed to extend a student's understanding, requiring a system for social interaction.

The resulting components of the learning system should be placed in a more technical framework. The placement of these components in such a framework is the subject of chapter four. From the above mentioned aspects, it can be concluded that especially the retrieval component of a general Information Retrieval system is important. The following chapter focuses on this aspect, and describes it in detail using Client/Server based Information Retrieval, Intelligent Tutoring System and Expert System theory.

3. Components of a Distance Learning system

3.1 Introduction

From the previous chapters it can be concluded that the system needed for this project should include a certain amount of intelligence, in the form of automatically adjusting the environment to an individual student. The distance learning system as it should be implemented is based on the World Wide Web, a client/server based information retrieval and presentation environment. Therefore, the model presented in this chapter is based on a functional description of the World Wide Web. Here, the needed intelligence should primarily be implemented in the retrieval component, which is practically non-existent in a standard World Wide Web environment. This chapter therefore focuses on a way in which the retrieval component of the system should be set up. In standard Expert System theory the detection and correction of deficiencies in students' understanding of a subject domain is presented as an area in which Expert Systems can be applied. It can therefore be expected that the retrieval component can be based on Expert System technology. Such an approach can be found in the domain of Intelligent Tutoring Systems (ITS) or as it is also called Intelligent Computer Aided Instruction (ICAI). This chapter thus moves from a functional model of the World Wide Web in terms of client/server based information retrieval to a model for distance learning by looking at Intelligent Tutoring Systems and Expert System theory and practice.

3.2 Client/Server based Information Retrieval

The World Wide Web is basically a client/server based information retrieval system, having both Presentation Services as well as Presentation Logic located in the Web browser. It is a two-tier architecture having client and server communicating using synchronous Remote Procedure Calls (RPC), where a client calls a function on a remote server and waits for the result before continuing its processing (actually, a number of such calls is done at once). Furthermore, such communication is stateless, meaning that no state information is preserved between sessions (a session consists of a client request followed by a reply from the server). Preserving state can be done by storing session information either at the server or the client (using so called 'cookies'). To allow a client to communicate with processes other than a Web-server, the Common Gateway Interface (CGI) has been developed. Using CGI, a client still connects to a Web server, but this server starts a process of which the output is send to the client. This process can connect to other processes, making it possible to build three-tier or even multiple-tier architectures. In chapter 6, the finer details of the World Wide Web environment are described, as far as they are important for the model described in this chapter. Information retrieval on the World Wide Web is basically half-duplex in functionality and based on the pull model (where the activator is the destination of the information). However, the World Wide Web currently also provides means for implementation of the push-model (see for instance (Yeh, 1996)). Information can be made available to everyone (multicasting), or to a specific group of people (narrowcasting or even unicasting). The basic model which can be used to describe information retrieval systems is shown in Figure 3-2, and is based on the model of information retrieval found in (Ackerman, 1993). In this figure, the retrieval component interacts with both the information store as well as the user. From general descriptions of agents it is clear that they function on behalf of a user, using a model of that specific user's interests. In a learning system, the role of agents is one of modeling a user's current knowledge, to request the right information from the system to enhance a user's

knowledge. Based on knowledge about the student's state, agents decide on the next piece of information necessary for the learning of the student. Agents here are internal to the system, instead of external. Next, the agent's request should be translated to the retrieval of a certain presentation of subject matter. Finally, the subject matter is presented to the user.

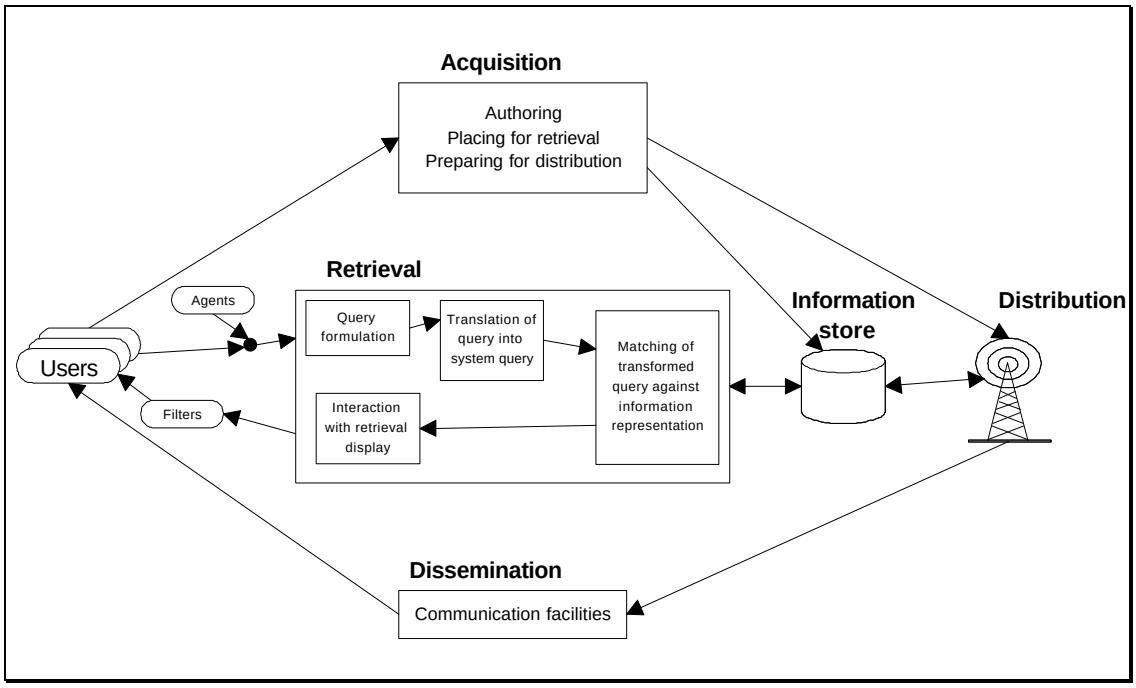


Figure 3-2 - Information retrieval model, based on (Ackerman, 1993)

Filters as they are generally described are not present in the system, since they assume that the user wants to filter out certain information. The important components therefore are: an extended agent, a retrieval component and a storage component. The implementation of these components as they are found in Intelligent Tutoring Systems is presented next.

3.3 Components of Intelligent Tutoring Systems

The field of Intelligent Tutoring Systems (ITS), focuses on problems of knowledge representation, student misconceptions, and inferencing (Kearsly, 1987). The design and development of such systems lies at the intersection of computer science, cognitive psychology, and educational research. Traditionally, five ICAI paradigms are identified (Kearsly, 1987). In the *'Mixed Initiative'* paradigm, the program tries to teach a student by engaging him or her in a two-way conversation based on natural language (a 'Socratic dialogue'). The *'coaches'* paradigm is aimed at observing student performance and providing advice to help a student perform better. *The 'diagnostic tutor'* approach uses a so called 'bug catalog' to identify misconceptions students may have in solving a problem. The *'microworld'* paradigm allows students to explore a problem domain by creating a stimulating graphical environment. Finally, the *'articulate'* paradigm involves the use of expert systems able to explain their decisions. Due to the fact that the World Wide Web is primarily a sequential presentation medium which is event-driven, it can be concluded that it is not best suited for the on-going interaction of the 'Mixed Initiative' approach or the creation of a complex microworld. And since the Artificial Intelligence solutions for the creation of a diagnostic tutor are neither generally available nor suited for the focus of this thesis, the coaches paradigm and

the articulate expert system approaches are used here. Traditionally, such systems are similar in the fact that their implementations share four components:
1. Domain Knowledge component;
2. Teaching component;
3. Student Model component;
4. Interface component.

This division is based on presenting both the student and the system with a flexibility in the learning environment which closely resembles what actually occurs when student and teacher sit down one-on-one and attempt to teach and learn together (Park, 1987). The model interconnecting the several components is depicted in Figure 3-3.

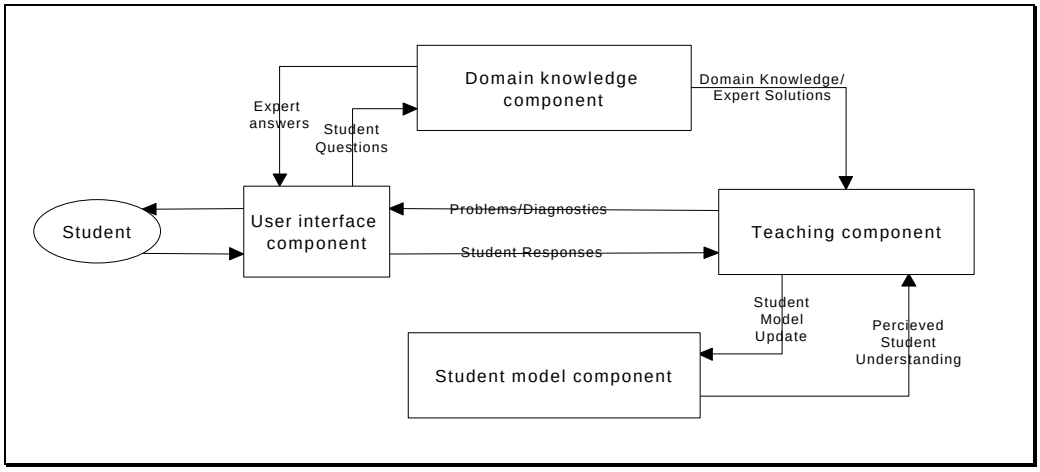


Figure 3-3 - Software components of an ITS, based on (Mast, 1995) and (Wallach, 1987)

Using such a separated structure, content or instructional decision rules can be modified without causing reformulations of the whole structure. However, comparing it to the information retrieval model in Figure 3-2, the Teaching Component seems to be responsible for at least three separate activities: deciding on the right piece of information (agent function), retrieval of this piece of information and interacting with the user interface. Therefore, this model merely adds a domain knowledge component and student model component to the basic model of information retrieval. It should therefore be seen as a description of how an agent should operate in a learning system.
The User Interface controls communication between the student and the system. Here, student input is analyzed and formulated into a machine usable code. The output of the system is presented to the student in a suitable form using guidelines for layout and interaction format (Mast, 1995) The other three components are generally seen as expert systems, and are described in the following sections.

3.3.1 Domain Knowledge component

This component is concerned with storing, manipulating and reasoning with knowledge of some subject domain, either to answer student questions or function as input to the Teaching Component. Artificial Intelligence methods used to organize knowledge in this module include development of semantic networks, application of production systems, procedural representations and building of script frames (Park, 1987). These methods can broadly be categorized into usage of a model-based or curriculum-based structure (Khuwaja, 1996). Basically, a model-based component uses an algorithmic approach towards reasoning about the

domain, while a curriculum-based component uses knowledge about the way concepts are related to each other within a domain. The first approach is often chosen when the goal of the ITS is to replace a tutor by the system, which then has to reason about the domain to be taught in an intelligent way. In this thesis, the use of semantic networks in a curriculum-based approach is used to enable the Domain Knowledge component to support a real instructor. Chapter 4 focuses on this specific aspect.

3.3.2 Teaching component

The teaching component contains some simple or complex decision rules to determine the sequence and style of presenting subject matter and feedback to the student, it also determines if the subject is mastered or not (Mast, 1995). According to Mark and Greer, the teaching component of an ITS should be like an expert which designs and guides a tutorial session in accordance with instructional methods and conditions for their use (Mark, 1992). A goal of ITS developers is to build systems which will create individualized instruction, accommodating student characteristics such as background knowledge, level of cognitive development, and learning style (Mark, 1992). It is at this component that an ITS adapts a course to how well the student is performing according to the goals set for different parts of the course.

3.3.3 Student Model component

In (Park, 1987) it is said that methods to apply student modeling include both quantitative as well as qualitative approaches. The quantitative approaches are based on charting the areas which the student has mastered or has attempted to learn. In (Park, 1987), two models of constructing a student model are identified. The first is called the *'overlay model'*, where a student's knowledge state is represented as a subset of an expert's knowledge base. The student model is then constructed by comparing the student's performance to the computer-based expert's behaviour on the same task or problem. The other model is called the *'buggy model'*, representing the student's mislearned subskills. These subskills are not subsets of the expert's knowledge, but are seen as being variants of the expert's knowledge. One often finds the use of the Bayesian probability theorem as part of such student modeling. A more complex approach is to base the student model on qualitative factors, where students' learning is assessed from the analysis of their responses or response patterns, using pattern recognition. Both approaches have their limitations. For instance, quantitative methods require sophisticated routines to assess student's knowledge at higher levels while qualitative methods rely heavily on the developer's subjective judgment instead of objective criteria (Park, 1987). A combination of both might present the right solution for problems of assessment at all knowledge levels. In chapter 4, a quantitative method for student modeling is described.

3.3.4 Conclusions

The basic model used in Intelligent Tutoring System theory can be seen as an extension to the model of information retrieval of Figure 3-2 in that it described how the agent function can be carried out. What is missing in such a model is first of all a clearer distinction between the different functions found in the teaching component. Second, the focus on excluding the instructor from the educational process results in an unrealistic presentation of the autonomy of the system: in reality it needs an expert to provide and update the information present in the several components. The isolation of a student is clear in the model, since no provision is made for providing social interaction. Finally, when such systems are compared with Expert Systems, both knowledge refinement and explanation subsystems are missing.

3.4 Expert System theory

Not only can each component be seen as an expert system, the system as a whole can also be viewed from the perspective of expert systems. However, missing from the model presented so far are an 'Explanation Facility' and a 'Recommended Action' interface commonly found in Expert Systems. The last one informs the user of the conclusions drawn by the Inference Engine, while the first interacts with the user to explain the details of the decision. Both are aimed at providing the necessary interaction for the 'articulate' paradigm described above. Next to this, an Expert System usually contains a 'knowledge refining system', enabling the system to learn from its mistakes. Both the expert as well as the system can provide input to the knowledge refinement module, resulting in a more dynamic system. Literature refers to at least three different possibilities to include such knowledge refinement in an ITS:

- Process evaluation based on test-results as described in chapter 1. Test-results can be used to examine how the educational process can be improved, since problems in answering questions might imply problems with the educational process (Dousma, 1995);
- Solutions presented by students can become part of the system, to show future students where other solutions have failed or succeeded. This is generally used in systems using case-based reasoning, such as the one presented in (Fuji, 1996);
- In (Brailsford, 1997) the previous approach has been turned around: expert answers to student questions are added to knowledge already present in the system.

This last approach is a way of implementing a student-tutor discussion facility. In chapter 4, both the first as well as the last approach are described in more detail. The system described in this thesis does not focus on case-based reasoning, although in a future implementation this could be added to the practical assignment component. The role of an instructor in this system is one of providing knowledge about the specific domain, and helping in refining the Knowledge Base. His or her role is however not directly noticeable for a user.

3.5 Extending the basic model further

Some of the more modern approaches towards creating an ITS try to incorporate the specific possibilities of the Internet. Often, the World Wide Web is chosen as a platform. Since the World Wide Web is by definition a presentation medium, and not really suited for a Socratic dialogue, a combination of ITS and WWW usually moves into the direction of an intelligent system, guiding the student through the amount of information available about a certain domain. Next to this, the World Wide Web presents possibilities for interaction not only between a student and the system, but also between students and instructors. Thus, the traditional ITS approach of trying to replace an expert is partially abandoned in favor of an approach where experts have a place in supporting the student in his or her travels through the broad information space. Such a system can for instance take the form of an intelligent guide, with the presence of a human expert to provide knowledge not (yet) present in the system. A description of an intelligent guide not incorporating human experts can be found in (Khuwaja, 1996). Figure 3-4 presents the architecture of this intelligent guide.

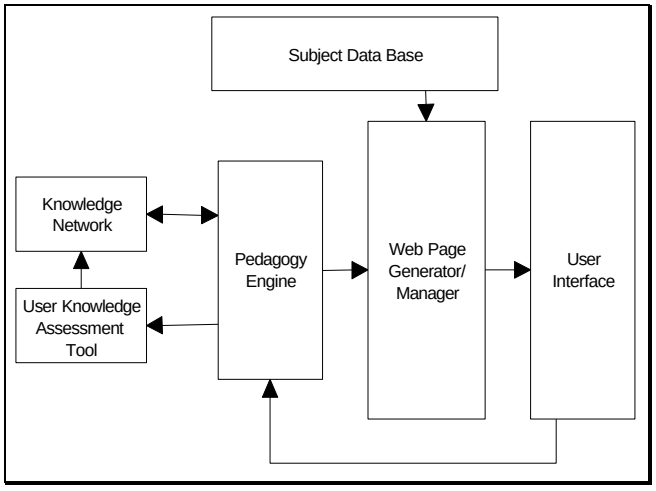


Figure 3-4 - Architecture of Intelligent Guide, (Khuwaja, 1996)

The systems itself is based on presenting domain knowledge in the form of a curriculum structure. The major objective of the Intelligent Guide system is to provide guidance to the user in the pursuit of achieving a satisfactory level of competence in a domain (Khuwaja, 1996). Based upon assessment of the student's knowledge, the system directs a student to areas in the knowledge domain requiring his or her attention. What it basically adds to the models already presented is a 'Web Page generator/manager' module, placed between the pedagogy engine and the user interface. Connected to this module is the 'Subject data base', which consists of the actual content of domain entities. The Web Page module does not interact with the user, it merely presents the information in a format readable by Web browsers. The two modules present WWW-based implementations of the retrieval and storage component mentioned in the first paragraph. However, the student communicates directly with the Pedagogy Engine. This way, the Pedagogy Engine should understand the input provided by the student, which is in a format specific to the World Wide Web. To enable the pedagogic part of the system to operate independent of the presentation environment, a new component should be placed between the User Interface and the Pedagogy Engine, translating student input to a Pedagogy Engine readable format.

3.6 A model for a Distance Learning system

From the description of the components in the educational process, it becomes clear that a student model can be used to improve the educational process, and thus the educational software. In the discussion about 'modellers' and 'non-modellers' in (Wasson, 1996), it is mentioned that this can either be done by the system itself, or by the instructor. Next to this, since an instructor is available, he or she can directly answer a student's questions, instead of implementing this into the Domain Knowledge component. Therefore, the following components should be added to the basic model: a learning component in the form of an instructor, an answer component which is an intermediary between student, and instructor and intermediaries to separate the pedagogical components from the presentation components. Next to this, a social component could be explicitly added to the model. This kind of social interaction is usually embedded in systems to provide interactions between peers and between students and instructors on a more informal level. Finally, the different functions of each component are added to the model. This results in the model depicted in Figure 3-5.

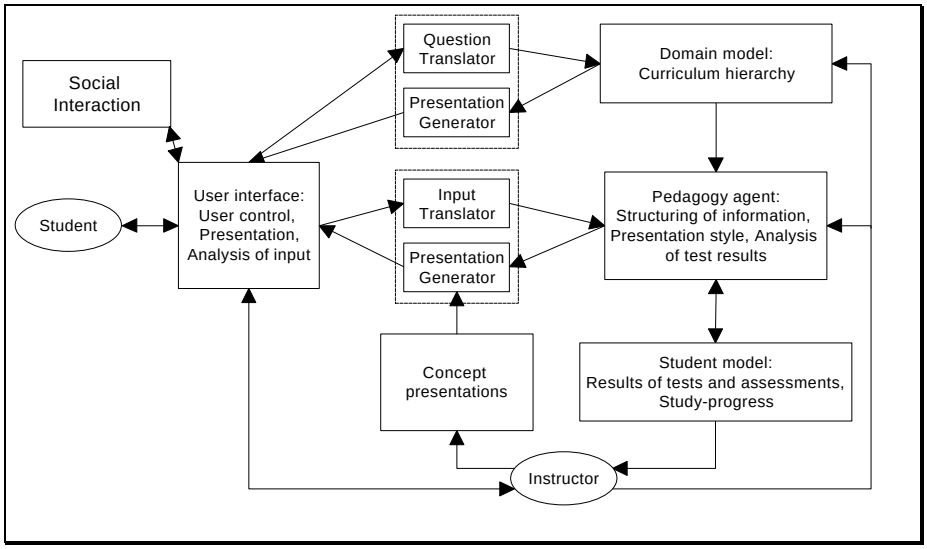


Figure 3-5 - Model for a Distance Learning system

In this model, the concept presentations component supports the aspect of providing a student with different presentations of the same concept. The pedagogy agent decides on which presentation should be presented. The intermediaries between the pedagogic components and the user interface enable a true client/server system based on presentation independence. In this model, two information retrieval systems can be identified: one based on providing a student with expert answers to his or her questions, an other to provide a student with the right presentation of concepts. The theory underlying these two systems is the subject of chapter 4, while the way they can be connected to a World Wide Web environment is described in chapter 5. Chapters 6 focuses on existing solutions on the World Wide Web, and chapter 7 describes the implementation of a prototype of the Virtual Computer Lab as an example of a concept presentation.

4. Domain knowledge and student modeling

4.1 Introduction

The basic model as it can be found in chapter 3 describes a presentation independent part consisting of a domain knowledge component, a student model and a pedagogic component. This chapter describes the theories underlying the implementation of these components. CERFACS, one of the partners in HPCnet, seems to have been researching the use of a qualitative approach towards student modeling based on presenting a limited course to a student. The final course there is created based on three types of data: the path a student takes through the course, the need for help when viewing a concept and a student's request for more information on specific concepts. In this thesis, a quantitative approach towards pre-testing a student's knowledge is described, including the limitations of such an approach. The last part of this chapter describes a way to implement a structured question/answer system, combining domain knowledge and the presence of both peers as well as experts. Both systems use the Domain Knowledge component, which is based on a structure described next.

4.2 Structuring a course

There is a difference between the sequential structure of a course as it is presented to a student, and the conceptual structure describing the way concepts are related to each other within a domain (as was presented in chapter 2). The structure used in this thesis is what is called the directed acyclic graph of a curriculum hierarchy, and uses the overlay method. Within such a structure, two relationships can be important: abstraction and aggregation. This is basically similar to using ISA and Part-Of relationships. Most courses can be modeled using an aggregation structure, including the course which is presented as a case study in this thesis. Within such a structure, AND and OR relationships exist, as well as so called prerequisite relations (Collins, 1996). AND relationships can be used to describe a group of concepts which are all needed to understand a higher level concept. OR relationships represent different views of the same concept, where understanding one of these views is enough to understand the concept itself. Prerequisite relationships between concepts can be used to guide test ordering (Collins, 1996). However, for the discussion about using Bayesian networks and testing theory prerequisite links as Collins describes them do not appear. It is further assumed that the student makes the choice when an OR relationship is encountered, based on his or her knowledge or interests. The focus in this chapter is therefore primarily on applying Bayesian probability theory to Part-Of structures. Such a structure could for instance be created using a (simple) SQL-Database, consisting of conceptual information and links between concepts. Each link is then a separate entity, containing a from-field, a to-field, a direction and a type.

4.3 Fundamentals of quantitative computer based adaptive testing

4.3.1 Limitations of quantitative student modeling

Quantitative student modeling is based on using the computer to decide on a student's performance. The criteria and demands such tests have to meet can be divided into three levels: the test as a whole, the items a test consists of and the alternatives present in such a test-item. The demands and criteria at the test level can be summarized into four categories (Dousma, 1995):

1. Efficiency: this is seen in terms of the time it takes an instructor to construct and correct tests and the time it takes a student to finish the test, compared to the information a test produces
2. Objectivity: are the questions and answers clear, thus minimizing the influence of the person who corrects the tests?
3. Reliability: to what extent does the test provide consistent results, regardless of the goal
4. Balance: does the number of questions, and level of questioning, correspond with the goals the instructor had in mind when creating the test?

In (Dousma, 1995), different methods of testing have been compared with respect to the four goals and four criteria mentioned above. This has resulted in Table 1.

+ = well suited for use +/- = more or less suited - = not suited for use	Oral tests	Essay tests	Short answer tests	Completion tests	Multiple choice tests	Yes/No tests	Matching tests
Efficiency	-	+/-	+	+	+	+	+
Objectivity	-	-	+/-	+	+	+	+
Reliability	-	+/-	+/-	+	+	+	+/-
Balance	-	-	+/-	+	+	+	+
Think questions	+	+	+	-	+/-	-	+/-
Application questions	+	+	+	+/-	+/-	+/-	+/-
Comprehension questions	+	+	+	+	+	+	+
Knowledge questions	+/-	+/-	+	+	+	+	+
Comments	**Small groups, special situations**	**Relatively small number of students**	**Around 40 students**	**Large groups, or repeating use**	**Large groups, repeating use**	**Subject matter specific, around 40 students**	**One error results in more errors.**
	Open Questions				**Closed Questions**		

Table 1 - methods of testing compared with respect to goals and criteria, (Dousma, 1995)

Higher abilities are the hardest to test on a computer, if they can be tested at all. Yet, by comparing coursework, written exams and computer exams with respect to the five categories of learning presented in chapter 1, in (Cellear, 1997) it is concluded that by combining coursework and computer exams, one can assess everything one might want to know. And this can be done without using written (or oral) exams. So, theoretically, combining computer exams and coursework creates an environment in which the four criteria of reliability, balance, objectivity and efficiency can be met, while still reaching the goals of assessing higher abilities and additional skills. Such assessment can be used in a predictive, selective or diagnostic way, and is important for creating a student model which can be used by the system, the student or the instructor. The next part of this chapter focuses on the use of adaptive testing algorithms in Intelligent Tutoring Systems.

4.3.2 Test item analysis

To describe adaptive testing, six different steps can be identified (Hambleton, 1991): choice of an item response model, creation of an item bank, deciding on a starting point for testing, selection of subsequent test items, scoring/ability estimation and deciding on termination criteria for the test. The theories described in this paragraph can be used to ensure quality of the items in the item bank.

In (Dousma, 1995), an extensive description of test item analysis can be found. Such analysis focuses on level of difficulty, quality of alternatives and discriminative power of a test item described in the following formula's:

- General level of difficulty: *p=number of students answering item correctly/total number of students*. Since this value does not take guessing into consideration, the level of difficulty corrected for guessing is defined as: *p'=p-(1-p)/(na-1)*, where *na* represents the number of alternatives per item;
- Attractiveness of an alternative: *a=number of students choosing the alternative/total number of students*. The *a* value of an item should not approximate or be equal to zero, and should always be smaller than *p'*. If not, either the item is incorrect, or the course does not pay enough attention to how such a question should be answered. All *a*-values of an item should be approximately equal, so each alternative functions the way it is supposed to;
- Quality of alternatives: $Z_a=([X_a]-[X])/S_x$, where $[X_a]$ equals the mean score for students choosing the alternative and [X] equals the mean score of all students. S_x is the standard deviation of all test-scores. The quality of a wrong alternative should be negative, and the quality of the right answer should be positive. This way, the item can distinguish good students from bad students;
- The discriminating power of an item, which should be high so that it can distinguish good students from bad students. Here, only the simple calculation is provided: select two equally sized subgroups of students, one representing the highest overall scores, the other representing the lowest overall scores. Identify the number of students answering the item correctly in both groups. Divide both by the size of the group, and then subtract the number for the last group from the number for the first group. According to (Dousma, 1995), choosing 27% of the total group of students for both sub-groups provides the best results.

The theory described in (Dousma, 1995) is therefore useable to ensure quality of test-items within an adaptive testing environment. The actual model used to decide on the right questions to ask a student means choosing a specific implementation of the Item Response Theory (IRT). This theory is based on the idea that a student's answer to a question presents a certain probability of that student mastering the specific concept. Therefore, scoring and ability estimation as well as test termination is based on the results of a number of tests, representing a subset of all the tests available. In standard Artificial Intelligence literature, the use of Bayesian Belief Networks for such reasoning is advocated. The following two paragraphs describe the IRT and Bayesian Belief Network theories.

4.3.3 Item Response Theory

The Item Response Theory is aimed at providing information about the functional relation between an estimate of a student's mastery level of a concept (his or her 'latent construct') and the likelihood that a given respondent will make a correct response to a given item (the Probability of Correct Response or PCR). In general, three models are identified based on using the difficulty parameter, the discrimination level and the guessing factor of a test item: the so called 'Rasch model' using only the difficulty parameter, the 2-parameters model using both the difficulty parameter and the discrimination parameter and the 3-parameter model. According to Hambleton et al., this last model is most appropriate in adaptive testing (Hambleton, 1991).

Using IRT, each item on a test has its own characteristic curve (ICC), which represents the probability that the student will be able to provide an answer which will enhance the believe the

student is a master. The mathematical expression for the three-parameter model is (Hambleton, 1991):

$$P_i(q) = c_i + (1-c_i)\frac{e^{Da_i(q-b_i)}}{1+e^{Da_i(q-b_i)}} \qquad i = 1,2,...,n$$

Here, $P_i(\theta)$ is the probability that a student with ability θ answers item i correctly, depicted as an S-shaped curve with values between 0 and 1 over the ability scale. The difficulty parameter for item i is depicted as b_i, the item discrimination parameter as a_i and the pseudo-chance-level as c_i. This last parameter takes into account performance at the low end of the ability continuum, in terms of guessing. The number of items in the test is given by n, and the factor D is a scaling factor usually approximately 1.7. Using this formula, the Item Information Function (IIF) can be calculated. This is a representation of the amount of 'information' provided by each item at any given level of latent construct. Information in IRT terminology describes how well the item can discriminate between individuals with different latent constructs. The general equation as well as the one used for the three-parameter model are given below.

$$I_i(q) = \frac{[P'_i(q)]^2}{P_i(q)Q_i(q)} \qquad i = 1,2,...,n$$

Equation a - Item Information

$$I_i(q) = \frac{2.89a_i^2(1-c_i)}{\left[c_i + e^{1.7a_i(q-b_i)}\right]\left[1+e^{-1.7a_i(q-b_i)}\right]^2}$$

Equation b - Three-parameter model

Here, $Q_i(\theta)$ is equal to $1-P_i(\theta)$. Using this information function, for each latent construct the item providing the most information can be chosen to increase probability of a student being either a master or a non-master. To propagate this knowledge throughout the course hierarchy, Bayesian Belief Networks can be used.

4.3.4 Bayesian Belief Networks

Many ITS systems use Bayesian network theory to update believe about a student's knowledge. In the following two pictures, the situation is shown for two ways to present an environment where two questions are used to determine whether a student masters a subject or not.

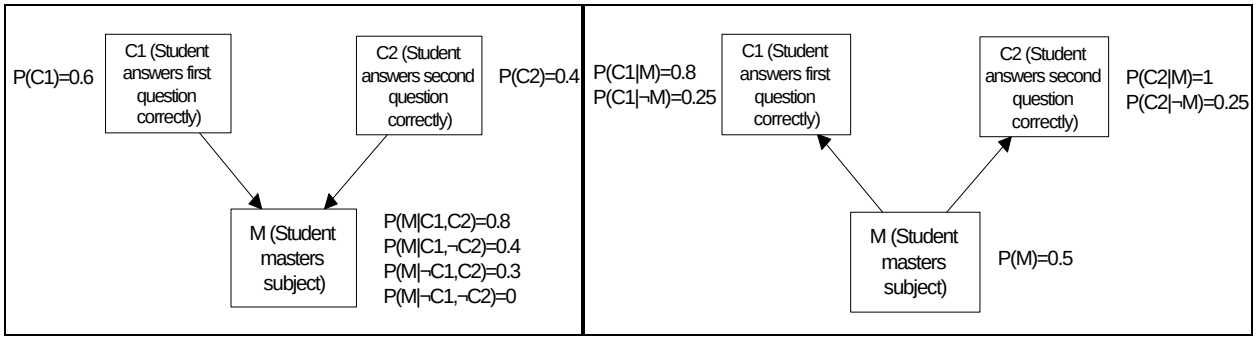


Figure 4-1 - Two ways of representing a Bayesian Belief Network

If an instructor must provide the different probabilities based on experience, the second representation is clearly preferable from an efficiency point of view. This leaves the problem of deciding on P(M|C1,C2), P(M|C1,¬C2),P(M|¬C1,C2) and P(M|¬C1,¬C2).

For this, Bayesian nets can be used. Basically, this theory states:

$$P(A,B,C,...,Z) = P(A \mid B,C,...,Z)P(B \mid C,...,Z)...P(Z)$$

Combining this with standard Bayesian probability theory, the following calculation describes the application of this rule to the example above:

Translating the rule to the situation in the example, it states: *P(M|C1,C2)=P(M,C1,C2)/(P(C1|C2)P(C2))*
Now, since C2 does not influence C1, the following is true: *P(M|C1,C2)=P(M,C1,C2)/(P(C1)P(C2))*
And, reapplying the rules to P(M,C1,C2), the final result states: *P(M|C1,C2)=P(C1|M)P(C2|M)P(M)/(P(C1)P(C2))*
Since for both P(C1) and P(C2) the following rule is true: *P(C)=P(C,M)+P(C,¬M)*
the final formula states: *P(M|C1,C2)=P(C1|M)P(C2|M)P(M)/(((P(C1|M)P(M))+(P(C1|¬M)P(¬M)))(((P(C2|M)P(M))+(P(C2|¬M)P(¬M)))*

This is basically how Bayesian networks are used in adaptive testing: based on information about answers to questions belonging to a certain concept, the probability that a student is either a master or a non-master at that concept is calculated. One of the drawbacks of use Bayesian nets is known as the 'optimal factoring problem'. With the transformations presented above as an example, the problem comes down to choosing the right order of events. If P(C1,M,C2) was transformed instead of P(C1,C2,M), at least four extra steps would be needed. For calculating belief in larger networks, several algorithms targeting such problems have been developed. In general, finding the optimal factoring is considered to be an NP-hard problem, but a polynomial time algorithm exists for generating optimal factoring for tree-structured belief networks such as the one used here (Ambrosio, 1991). This theory, and algorithms to use can be found in for example (Ambrosio, 1991).

4.3.5 Combining test analysis and Bayesian Belief Networks

When the rules of item-analysis are applied to the situation where a Bayesian network is used to determine mastery, a basic adaptive testing system aimed at deciding on mastery can be described. In such a system, the item pool should be created by an expert and the entry level of a student should be provided by either the student or the instructor. Basically, the probability of a student being a master can be compared to two threshold values: if it is smaller than the lowest threshold non-mastery is decided, if it is larger than the highest mastery is decided. If no decision can be made, a new question will be asked until there is enough evidence to make a decision. Therefore, for each item the right answer, the number of alternatives, the number of students choosing a specific alternative and information about a student who has answered the question being a master or non-master is needed.

From the above description, the following conclusions can be drawn:

- This approach requires extensive calibration of the system. Many values are dependent on knowledge about the number of students or the number of masters answering the specific item;
- The system can become a learning system if it keeps refining its values based on student responses;
- After a student has provided the system with a temporary entry-level in the form of a concept, the system should decide which concepts are mastered by the student. However, the total number of questions should be limited, for a long list of questions will bore the student.

There are various systems implementing the theory described above. For this project, the Content-Balanced Adaptive Testing system (CBAT-2) described in (Huang, 1996) seems useful. This approach extends the theory described by Frick and Welch who resolved the need of the traditional Weiss/Kingsburry IRT implementation for extensive calibration.

4.3.6 The Content-Balanced Adaptive Testing system

The CBAT-2 system, described in (Huang, 1996), aims at providing a solution for the following problems not addressed by the Frick and Welch approach: content balance, intelligent selection of test items, security so that selected test items do not form a pattern, possibility of associating questions with multiple content areas and the notion of two-level assessment as will be described below. The Domain Knowledge component in this system consists of the directed acyclic graph of a curriculum hierarchy. As described before in this chapter, this graph associates content areas and questions in a course as is shown in Figure 4-2.

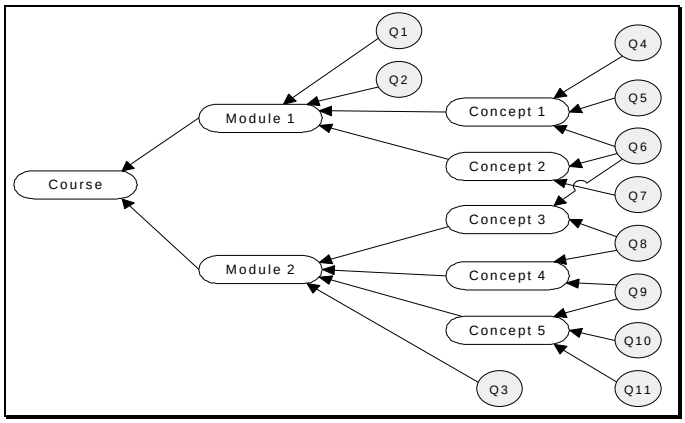


Figure 4-2 - Curriculum hierarchy in CBAT-2, (Huang, 1996)

In this hierarchy, questions associated with the lower levels require knowledge of specific concepts and skills. Questions at higher levels require general knowledge. The system uses a two-levels approach towards both content balancing and assessment at different levels. This is based on the fact that mastery at a higher level means mastery at all lower levels which are prerequisite for the higher level. This has at least three consequences:

- By repeating this approach, the knowledge level at each of the components involved can be tested;
- The test list will not become too large;
- Test from lower levels can be used at higher levels.

If for instance the system starts evaluating a student's knowledge at the course level, the system assesses knowledge at both the course level and the module level. If the system evaluates a student at module level, it assesses a student at both module and concept level. This approach is depicted in Figure 4-3.

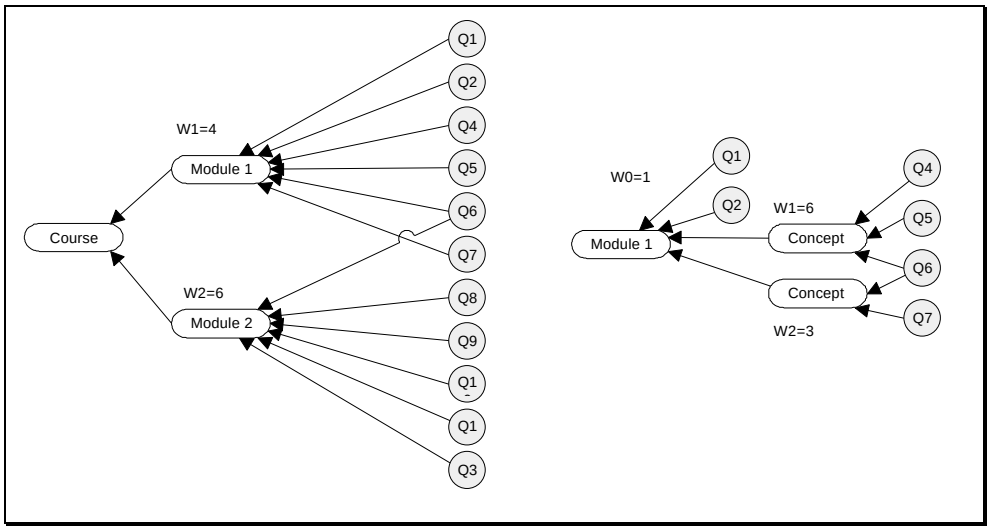


Figure 4-3 - Curriculum hierarchy for a course and module test, (Huang, 1996)

According to Huang, the discriminatory level is usually difficult to calibrate and its meaning is difficult to be understood by test designers, it is therefore defined as a constant with value 1.2. The guessing factor is simply defined as one divided by the number of alternatives. To calculate the difficulty level of a question, a more complex formula is used:

$$diff_i = \frac{20 \cdot init_i + \Phi_i}{20 + R_i + W_i}$$

Here, $init_i$ is the initial difficulty of the question, R_i and W_i are the number of times item i was answered correctly or incorrectly respectively. The constant 20 is used as a normalization factor. The system has a learning ability since Φ_i is defined as:

$$\Phi_i = \sum_{j=1}^{n} k_j \cdot f(q_j')$$

where n is the number of answers for item i in the past, θ_j' was the temporary proficiency of the student who gave the jth answer for Qj; k_j=0 if the jth answer is correct, and k_j=2 if the jth answer is incorrect. The function f converts a θ value to a difficulty value and is currently linear. The test designer assigns the initial difficulty, which has a value from 0 to 1. Using this approach, it is no longer necessary to calibrate the system using empirical data. As the question is used in tests, the difficulty level converges to $\Phi_i/(R_i+W_i)$, thereby minimizing the influence of the initial difficulty.

Each student starts with an initial proficiency, and the first question is selected based on this proficiency. Based on empirical data, the system calculates new temporary proficiencies along with the confidence degree for the proficiency. The question selection procedure first decides on the right component, and than selects a question from this component. The choice for the component is based on weights of the components involved. However, if proficiency is already decided on one of the components, this component is not included in the candidate components. Selection of a question belonging to the component is based on the amount of information that a question may provide for the student's assessment as described by the IIF. A set of questions with the highest score become candidate questions, among which a question is randomly selected. This is done to prevent the test from becoming deterministic.

To decide on test termination, the likelihood ratio LR described by Frick and Welch can be used (Collins, 1996). This approach is Bayesian in nature, and is described as:

$$LR \leftarrow \frac{P_{om}\prod_{i=1}^{n} P(i \mid M)^s [1-P(i \mid M)]^f}{P_{on}\prod_{i=1}^{n} P(i \mid N)^s [1-P(i \mid N)]^f}$$

Here, s=1 if item i is answered correctly and s=0 otherwise, f=1 if item i is answered incorrectly and f=0 otherwise; n is the number of items in the item pool, i is the event that a right answer is provided, M is the event of a student being a master and N the event of a student being a non-master; P_{om} and P_{on} represent the student's prior master probability and non-master probability respectively.

The test then terminates when two conditions are met:

- LR has passed an upper confidence criterion or LR has passed a lower confidence criterion;

- every child components has at least a predefined minimum number of questions selected for the test.

After termination, the temporary proficiency of each component becomes its proficiency. Based on this proficiency mastery or non-mastery can be decided.

Comparing CBAT-2 to theory previously described, three assumptions in the system may need further development. First, by not including a discriminatory level in the algorithm, previously described problems might occur in the testing environment. If the goal is to measure the student's ability to answer difficult questions, the algorithm is useful. However, if the goal is to decide on mastery or non-mastery, a discriminatory level should be included to be able to choose which question can best be used to decide on this.
Second, the guessing factor is highly dependent on the quality of the alternatives. The initial value could be set as is done in the CBAT-2 system, but as empirical results become available, the guessing factor should take the quality of alternatives into account. This is also mentioned in (Hambleton, 1991), where instead of 'guessing factor' the term 'pseudo-chance-level parameter' is used. The pseudo-chance-level parameter described the portion of lowest-scoring students answering a question right, and is in general lower than the guessing factor used in CBAT-2.
Finally, in CBAT-2, the preliminary proficiency of a student is used in the difficulty accumulator. However, this seems a bit peculiar: the preliminary proficiency is only an estimate of the final proficiency which is the result of testing the student. Practically, the preliminary proficiency doesn't tell much about the real student state, especially since no probability measure is used. In this case, it seems more sensible to use the final proficiency to update all questions asked. This is more complex, but seems to be more correct.

4.4 Providing expert help to students

As described in chapter 2, there is need for a system enabling students to gain help when they run into problems. For this thesis, such a system should be connected to the domain knowledge component as described in the system model in chapter 3. Ackerman has done a lot of research into a way to structure questions and answers in a dynamic, networked environment. In his Ph.D. thesis, he describes a system which aims at augmenting organizational memory (Ackerman, 1993). This system, the 'Answer Garden', is seen as being very useful for a distance learning environment (see for example (Brailsford, 1997)).

4.4.1 Answer Garden

Answer Garden is focused on organizational memory which concerns itself with usually recent events and outcomes within an organizational context and for organizational purposes.
The system was aimed at situations in which there is a continual stream of questions, most of which recur frequently, but there are always some that are novel (Brailsford, 1997). This basic idea behind the system should therefore make the system also applicable to use in a Distance Learning environment.
The Answer Garden is based on putting a well-structured database of previously asked question along with their answers between the asker and the expert. When someone has a question, he or she first has to access the database. If the right question cannot be found, or the user is not satisfied with the answer, an e-mail message can be send to the Answer Garden from the current location in the tree. The system than automatically forwards this e-mail message to an expert belonging to that particular part of the tree. When the expert answers the

question, the question-answer pair is added to the tree and a copy is send to the user. The Answer Garden can thus be seen as a structured Frequently Asked Question list, which grows 'organically'.

4.4.2 Knowledge tree

In the Answer Garden approach, each student can only communicate with an expert and is isolated with respect to its fellow-students. The only sign of life it gets from them, are the questions they have asked in the past and which are available in the Answer Garden. The Knowledge Tree approach described in (Brailsford, 1997) adds an important peer-to-peer discussion facility to the Answer Garden. Next to this, it uses a course hierarchy to present the user with a structured way of navigating, enabling it to be included in the system presented in this thesis.

The systems itself consists of three identifiable modules:
1. A discussion module, called a 'Forum';
2. A Concept Thesaurus, providing an interface to the conceptual structure;
3. A Knowledgebase, consisting of all question-answer pairs.

These components are structured as shown in Figure 4-4. The Forum contains all the student-student interaction. When a student asks a new question in the Forum, he or she has to attach a concept to it, by using the Concept Thesaurus. The question, with the attached concept, is then emailed to an expert. When this expert answers the question, the question-answer pair is added to the Knowledgebase. Threads in the Forum which become obsolete are removed completely from the Forum. Knowledgebase articles are however permanent.

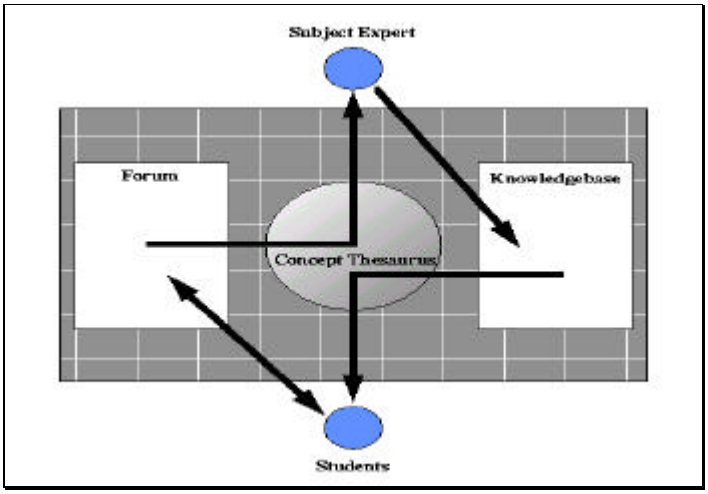


Figure 4-4 - The Knowledge Tree system model, (Brailsford, 1997)

4.4.3 Possible enhancements

Possible enhancements to the systems described above concentrate on three points:
1. When the Knowledge Tree becomes filled, problems will start to arise concerned with how quick a student can find the question he or she wants to ask. I therefore propose two adjustments:

- Connecting the position of a student within a course with its position within the Knowledge Tree. This can go as far as to sort the questions according to what questions are asked most often by other students at the same position;
- 'Weeding' out the questions which are hardly ever asked. Questions which are hardly ever looked at have no place in such an environment as this.

2. If the posed question or problem seems to be of a structural nature (for instance since it is asked by a large number of students), the instructor might change the course itself. By using an environment such as Knowledge Tree to enhance the quality of the course, the instructor does not have to wait until assessment results;
3. When discussing the system with instructors, they expressed the feeling that probably a student has to reach a certain level before being allowed to enter parts of the Knowledge Tree. For instance if part of the Knowledge Tree would be concerned with discussing several solutions to a problem, students still having to find a solution would not be allowed to view that discussion. Brailsford says that 'a good answer should promote reflection and further study rather than provide a package of self-contained information', but that does not seem to be a satisfactory solution.

Next to these three points, what is missing from the Knowledge Tree system model (Figure 4-4), is what Ackerman calls a 'QA-Tracker'. This system provides the necessary routines to make sure all question go to the right expert, and keeps track of those questions until they are answered. The resulting model for the system is shown in Figure 4-5.

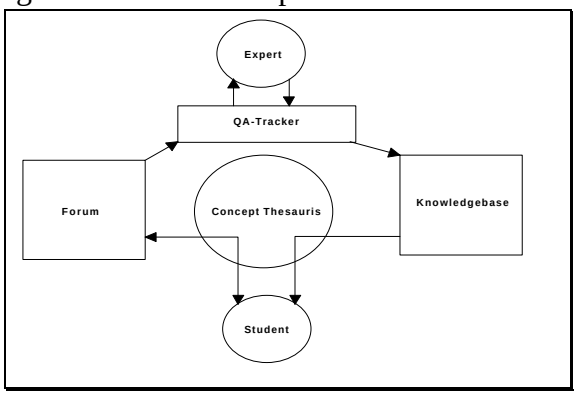


Figure 4-5 - Model for a changed Knowledge Tree

4.4.4 Help desk tracking systems

In a normal help desk situation, there is a need for so called 'help desk tracking systems' (named this way to distinguish them from the more programmer oriented 'problem tracking systems'). In general, such systems provide information on three different levels:

- Information about the one who asked the question (for example what question he or she has asked before);
- Tracking the state of the question (e.g. who is currently handling the question);
- Information about the experts (whether an expert is on vacation, etc.).

Normal help desk systems use a person as a so called 'first line'. This person is then responsible for assigning the questions to the proper experts. In the case of the system described here, it is the QA-Tracker who has to take care of assigning questions to the experts.

In (Ackerman, 1993), the following two tables are presented which describe the states of the QA-Tracker as well as special actions taken by the experts other than answering the question.

Question asked	User asks a question
Question forwarded	QA-Trackers send mail message to expert
Forwarded question accepted	Expert agrees to answer question
Answer returned from forwarded	Expert answers question
Question refused	Expert refuses the question (for any reason). QA-Tracker then goes to next expert if there is one
Cancel question	User cancels the question
Cancel request for answer	QA-Tracker cancels the request for an answer from the expert

Table 2 - QA-Tracker states, (Ackerman, 1993)

Question refused for time period	Expert refuses the question (for any reason) for a period of time. This might be used if the expert wanted to answer the question only if it were still outstanding after the specified period of time
All questions refused for time period	Expert takes himself off all expert lists for a specified time period. This allows the expert to adjust his workflow
Question refused but answer requested	Expert wishes to know the answer when one is available
Question forwarded by expert	Expert has forwarded the question to another person

Table 3 - Special actions taken by experts

When these tables are compared with the functionality offered by most help desk systems, one of the main things missing is a so called ‘escalation function’. This has as its primary responsibility checking whether a question remains unanswered for a period which is considered to be too long, and taking appropriate action to prevent this from happening. Ackerman himself therefore says two additional features are needed:

- The system should ‘nag’ experts to answer questions;
- The QA-Tracker must cancel questions when the expert has taken too long and must go onto the next expert.

In my opinion, such an escalation function should be defined at two levels: for the system as a whole, and for any question asked. The first is to ensure that none of the questions remain unanswered too long without anyone noticing, the second is to ensure that questions which should be answered within a certain amount of time are either answered or are marked as being ‘impossible to answer within the given time frame’. Both student and expert should be made aware of this fact, since this might have implications for the student’s progress.
Finally, in the event that a question cannot be assigned to any expert, the system could follow an approach similar to the one used for Internet Mail (the Simple Mail Transfer Protocol - SMTP): notifying both the administrator of the system as well as the student, and retrying for a specified period of time to assign the question to an expert. If this fails, the question is marked as being ‘undeliverable’.

4.4.5 Implementation aspects

If such a system would be build, each question could be represented by a database-entity, including the history of the question as it moves through the system. The relation with the conceptual structure is enforced by the student, resulting in knowledge about which expert to forward the question to. As long as the question remains unanswered, it is open for discussion. The moment an expert answers the question, it is placed in the Knowledgebase, closing it for further discussion. This enables the system to keep track of the number of students ‘asking’ that particular question, i.e. each student viewing the question and reading the answer. In a dynamic discussion environment such a connection does not exist. The system should be based on the usage of queues: one for the QA-Tracker and one for each expert. A student asking a question places it in the QA-Tracker queue. The QA-Tracker handles each entry in its queue according to Table 2. Forwarding a question to an expert means placing the question in the

queue of that particular expert, unless this expert has specifically stated to refuse all questions for a time period. The expert handles all questions in his or her queue one at a time, either accepting a question, answering a question or refusing a question. Such a queue could be ordered according to the escalation date of the questions. The status of expert queues can be used to evenly divide questions among available experts, creating a basic workflow management system.
Most actions follow a specific event initiated either by a student, the QA-Tracker or an expert, creating the need for a running QA-Tracker program. If an escalation function is included in the system, a prototype could make use of for instance a 'cron' job, checking the state of open questions at specified time-intervals. The event-driven nature of the interaction at both the student and the expert side make it likely that this can be done using a Web browser. Especially in the case of the expert, where the queue could be represented by a sequential list of questions, ordered by the server side of the system. Actions can than be initiated through the use of buttons. For the student, the list of unanswered questions can also be represented by a sequential list. The conceptual structure of the domain however, needs a different approach as will be described in chapter 7.
As a last remark, it should be obvious that the basic approach of adding questions to a specific concept can also be used to create a note-making facility, where a student can add personal comments to a concept.

4.5 Conclusion

In a model as described in the previous chapter, the conceptual structure of a course plays a central role. In this chapter, it was shown that by combining the overlay method with item analysis, item response and Bayesian Belief theories, a relatively complex adaptive testing environment can be created. Additionally, this environment has the capabilities to learn from the empirical data gathered by use in real situations. Any knowledge not readily available in the system can be gathered from experts, by allowing students to ask questions. Not only does this provide an implementation of the 'expert answer' part of the model in chapter 3, it also allows peers to communicate with each other in a structured way. A system in which students can attach questions to concepts can also be used for the much simpler requirement of a note-making facility. Although neither of the systems described here has been fully implemented for the prototype, the information should be useable for subsequent implementation phases.

5. Analysis of the environment

5.1 Introduction

The World Wide Web is a Client/Server environment and, with the emergence of WWW-browsers, a certain level of platform-independence is reached. Basically, movement is towards 'the browser as a platform'. In this thesis, the World Wide Web has been chosen as the environment for a Distance Learning system. This chapter focuses on the functionalities the World Wide Web offers with respect to presentation and communication. However, because of limitations of this environment, part of the Distance Learning system bypasses the protocols on which WWW is based. As said before, a Web browser has Presentation functionality, while Application Logic and Data Management are residing at the server side. A basic WWW system could therefore be described by the Thin Client/Fat Server type of system. However, with the addition of more intelligence at the client side, the movement is towards fatter clients. The system described in this thesis is mainly focused on a two-tier approach, which is inherent to the World Wide Web. However, the practical assignment environment described in chapter 7 can be seen as an approach towards a three-tier system.

5.2 The TCP/IP reference model

Although in (Hunt, 1992) it is said that there is no agreement about how to describe TCP/IP with a layered model such as the ISO/OSI model, both Tanenbaum and Hunt describe a similar layered model for TCP/IP. In these two TCP/IP reference models, the Data Link layer and the Physical layer of the ISO/OSI model are combined into one layer called "Network Access layer" (Hunt, 1992) or "Host-to-network" (Tanenbaum, 1996). In this thesis, I will not describe in detail layers one and two or the physical media involved, but as a generalization I will classify these connection methods as belonging either to a point-to-point network or a broadcast network. Figure 5-1 depicts the resulting model.

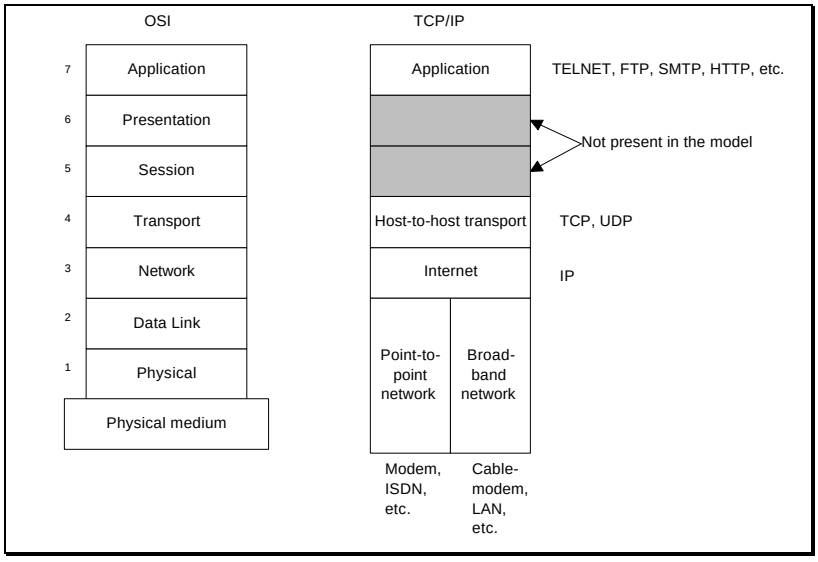


Figure 5-1 - TCP/IP reference model

5.3 Technologies involved in connecting the client to the server

Some of the more popular technologies currently in use connecting different parts of the Internet are depicted in Figure 5-2. Currently, a user at home is most likely to either have a modem or an ISDN (BRI) connection. Since most access providers have a digital connection with their local end office, new modem technologies connect the customer to their provider with a download connection speed of around 56 Kbps, exceeding Shannon's (theoretical) limit of approximately 35 Kbps on a standard telephone line. Asynchronous Digital Subscriber Line (ADSL) technologies and the use of Cable Modems are still in trial-phase. The use of ISDN (PRI) is primarily meant for business and Internet Access Providers to connect to the backbone. A T1 or T3 connection is a dedicated leased (physical) line, providing relatively high bandwidth, but can be very costly. ATM is currently being tested and being used at the national and international backbone level. In Europe, instead of T1 and T3, E1 (2.048M) and E3 (34.304M) are used.

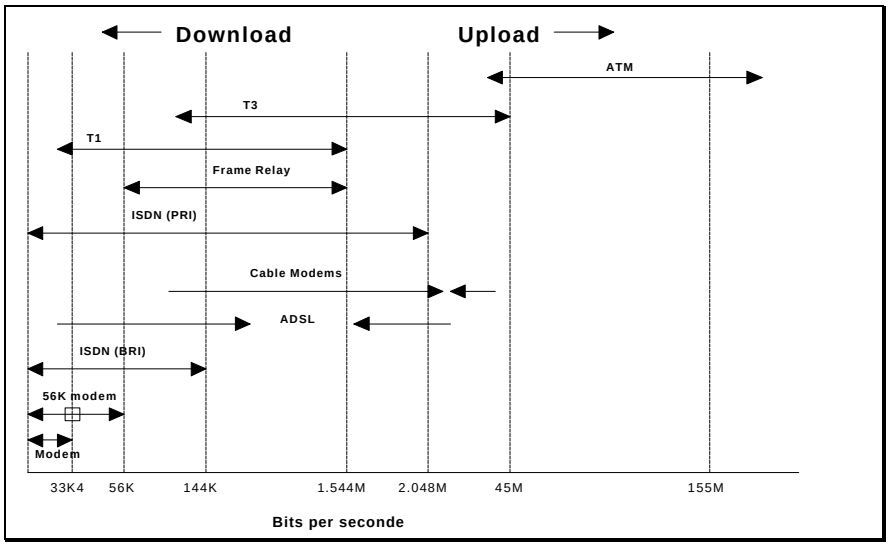


Figure 5-2 - Technologies for Internet connections

The Access Provider is connected to the national backbone using high speed lines between 2 Mbps and 34 Mbps. The national backbones are connected through a network of high speed connections (offering speeds of up to 155 Mbps). In Figure 5-3, the situation of the Ten-34 international (European) backbone is depicted. This network is an initiative connecting European institutions using a high-speed network. Its aim is to provide a complete 34 Mbps network. The different networks are connected through routers (or gateways). Currently, the 'home' of HPCnet is located at Southampton University, in England. The 6 Mbps Ten-34 connection was around 100 percent in use, the 16 Mbps connection around 40 percent and the 22 Mbps connection around 50 percent. Southampton University is connected to JANET (the national provider in the United Kingdom connected to Ten-34) via an SMDS connection. This results in the situation that data from the university is limited to 10 Mbps and data

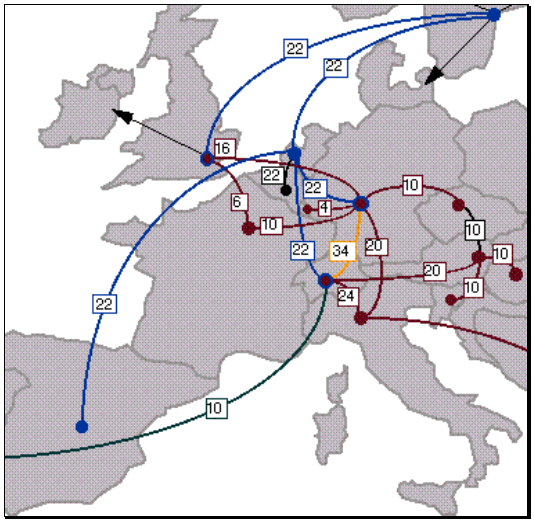


Figure 5-3 - The Ten-34 European backbone

to the university is limited to 34 Mbps (information provided by S. McDermid from JANET). In this case, only data from the university is important. If the student's situation is not considered, the 10 Mbps connection could become a bottleneck.

5.4 Prototyping a Distance Learning system on the World Wide Web

On the World Wide Web, the HyperText Transmission Protocol (HTTP) is used, but as will be seen, Java-applets can make use of the TCP-protocol directly. The creation of the prototype did not require any use of UDP, which is therefore not included in the description of the use of WWW. The basic configuration used for this thesis was as is shown in Figure 5-4.

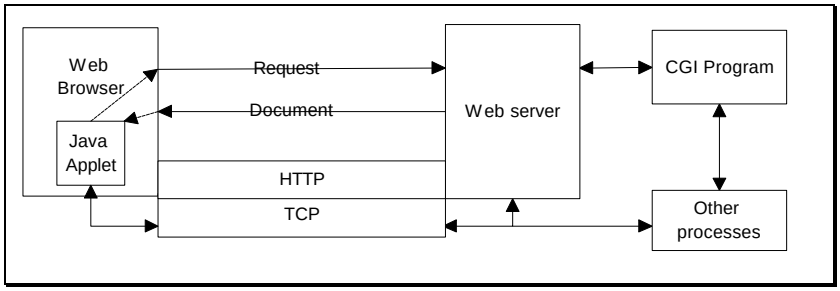


Figure 5-4 - Interaction between system components

This figure indicates that a Web-browser and a Web-server communicate through a request/response model, on top of HTTP. A Java applet can use the browser to communicate with a server (inheriting all of the browser's functionality), or communicate using TCP directly (although restricted by the server, as will be seen later). Additionally, clients can communicate with processes other than a Web server using the Common Gateway Interface (CGI). Such processes are referred to as 'CGI programs' and can be used to link a client to an application running either on the server machine itself, or on an other machine. In a large number of cases however, the CGI programs are the end-point of communication, providing functionality to the client. The test-environment used for this thesis was based on the type of hardware, software and protocols as described below.

- HTTP/1.0 was used, so neither the impossibilities of HTTP/0.9 nor the possibilities of HTTP/1.1 were included;
- Netscape Navigator 3.0x was used, thus only HTML 3.0 and the Java Development Kit (JDK) 1.02 were available for use. This means that the possibilities of HTML 4.0 or JDK1.x were not included in the results. Neither were the possibilities presented by other browsers (such as Microsoft Internet Explorer) included in the test environment. The choice for Netscape Navigator 3.0x was mainly based on the fact that during testing Netscape Navigator was the most complete client available for the Unix platform, and 3.0x was the newest version out of trial-phase;
- Although Netscape provides extensive possibilities for including plug-ins to enhance the browser, these were not included in this research. The main reason is that plug-ins for Navigator 3.0x are platform-dependent, meaning that they are (little) programs which run on the local machine. Including a plug-in for one platform, would exclude other platforms for which the plug-in is not available;
- Perl 5.0 was used;
- Various Web-servers were used, including a modified version of the Apache server and the Roxen Challenger server. CGI (Common Gateway Interface) and HTTP functionality included in the tests were the same for all servers;
- Finally, tests were done using high-speed bandwidth connections and basic 28K8 modem connections. No ISDN-connections were available at the time of testing.

Although it is possible that the functionality of a Distance Learning system can be degraded due to problems with firewalls and (caching) proxies, it is assumed that a student has access to the complete functionality of the system. The use of caching proxies is only considered in the light of possible bandwidth problems, described in the next chapter. Firewalls are shortly mentioned in the description of a system providing synchronized multimedia in appendix A.

5.5 Client side presentation functionality

5.5.1 Some comments on HTML

The HyperText Markup Language (HTML) of the World Wide Web can logically be divided into a markup language and a hypertext language. As a markup language, it is often described as being a (simplified) subset of the Standard Generalized Markup Language called SGML. As such, it offers a user the possibility to add a logical structure to a document, arranging it the way it is intended to be. It also offers a limited set of layout functions, but in general it is the Web browser which decides on the final layout which is presented to the viewer. Based on the fact that the HTML specification is constantly changing (version 4.0 is currently being developed) and the fact that HTML as a markup language is limited compared to for example SGML, the decision could be made to choose a markup language able to be translated into HTML or another markup language. This way, the system would become more flexible, both to future HTML developments as well as to moving the system to another markup platform.
The hyperlink capabilities of HTML are very limited. It basically provides a one directional link between an element in a document and an other document or a one directional link between an element of a document and another element of a document (possibly within the same document). Links are part of a document, so if the document is deleted, the link is also deleted. Removal of the document the link points to does not cause the link to be removed, since no back reference exists.
Links have no separate identity, have only one direction and the hypertext is not link-consistent as it is described in (Halasz, 1994). In a system as described in this thesis however, links should have a separate identity, including the possibility of back referencing. Link-consistency should be enforced to avoid such things as links pointing to removed documents. The problems generally identified in using hypertext (see for instance (Conklin, 1987)) have been weakened through the use of pre-structuring the environment for reasons described in chapters 2 and 3. Therefore, HTML is found to be suited at least for the prototype described here.

5.5.2 JavaScript

Client-side JavaScript is a language whose statements are embedded in an HTML page and can respond to user events such as mouse-clicks, form input, and page navigation. As can be seen, JavaScript is not included in Figure 5-4. The main reason for this is that network support is almost non-existent in JavaScript, and the main purpose of JavaScript is to extent the possibilities of Web page authors to control the browser environment in which their pages are shown. Therefore, JavaScript can be used to check user's input on the client side, without having to contact the server. In the prototype described in this thesis, JavaScript is hardly ever used. The few times it was used, was when other options did not exist to control the behaviour of the browser. Since primary focus here is on Client/Server communication, most focus is put on the use of Java applets and CGI programs.

5.6 The Common Gateway Interface

The Common Gateway Interface (CGI) is a rather simple way of creating the possibility for a client to connect to a running program either on the Web server machine or on another machine. A CGI program is a temporary process, started by a Web server and exiting after returning the desired output to the Web server, which sends it to the Web client. However, a CGI program does not need to connect to a running program. The information made available to a CGI program can be divided into three groups (based on (Gundavaram, 1996)):

- Information about the client, server and user (such as his or her username);

- Form data supplied by the user using the POST method;
- Additional information using the GET method.

CGI programs have the ability to function as the intermediary needed for the model of chapter 3, such that the pedagogical components can be made independent of the presentation environment. However, CGI programs follow the stateless architecture of a Web server, exiting after presenting the requested output. In situations resembling the Socratic dialogue situation, Message and Queuing (MQ) functionality is needed, where a reliable message path between applications is established. Only one type of MQ is considered in this thesis, namely process to process messaging, requiring both processes to be active. Next to this, the basic functionality of a Web client might not be sufficient, for example in situations where animations are required. Since the platform-dependency of plugins is undesirable, Java applets are considered in the last part of this chapter.

5.7 The HTTP/1.0 protocol for Client-Server connections

The HTTP protocol was aimed at providing the necessary network-functionality for distributed, collaborative, hypermedia information systems (Berners-Lee, 1996). Combined with HTML, each object within a document is requested separately. Problems previously occurring due to the slow-start TCP mechanism have been mostly resolved by the introduction of persistent connections in HTTP/1.0.
The functionality necessary for the system described in this thesis can be divided into four categories:

- Requesting and receiving standard documents from a server;
- Requesting and receiving dynamically created documents;
- Providing a way for the client to send data to the server, influencing the information returned from the server;
- Providing user-identification possibilities.

Each of these categories is in some way supported by the HTTP/1.0 protocol.

5.7.1 GET and POST methods

HTTP/1.0 implements three different methods for a client to request data from a server. The first one (GET), requests data identified by the used Uniform Resource Identifier (URI) which contains a scheme-identifier, a path to the requested document possibly containing query for a CGI program, and possibly a pointer to a certain position within that document. The GET method can also become a ‘conditional GET’, only returning the requested document if it has been modified since the date given by that field. To merely check whether a file exists, the HEAD method can be used, which returns only the document’s HTTP header.

To allow annotation, posting messages, providing a block of data to a data-handling process and appending information to a database, the POST method has been defined in HTTP/1.0 (Berners-Lee, 1996). When the POST method is used, a block of data is posted to the Web server, which is then usually send to a CGI program. Using forms, an HTML document can include an input area for a string of text, an input area for a block of text, a so called ‘radio button’ choice area, a checkbox choice area, a pulldown menu and a password field. When a user has filled in what is needed, he or she can press a button to transmit all information to a pre-specified CGI program. This action can be redirected to a JavaScript, which can check user input without first having to contact the server. The combination of forms and the various

methods to contact the server provides enough possibilities to use the World Wide Web in combination with the systems described in chapter 4. With the addition of the 'upload' facility available in versions 2 and higher of the Netscape Navigator, other media-types can be added to a plain-text message.

5.7.2 Authorization

A Web server can return a large number of status-codes after getting a request. Of these codes, the 'Client error' codes are of interest here. Client error code 401 (Unauthorized) is used to indicate to the browser that the user is not authorized to access the requested document. The user is generally asked by the browser to identify him- or herself using the following steps:

1. Browsers requests a page;
2. Server returns an 'Access denied' code;
3. Browser asks user to type in username and password, and adds this to the page request;
4. Server returns page;
5. Browser requests another page, and decides (based on the location of this page) to send the username and password along;
6. Server returns page.

In step 5, the browser makes the decision to send the username and password along based on the address of the server, and the location of the file. The 'Basic' authorization schema uses base64 to encode the username and password, which is a non-secure method. This schema is based on the assumption that the connection is a trusted one, which is definitely not true on the Internet. Also, it does not provide any mechanism for encoding the transmitted document. Some other protocols have been created to tackle these problems. These protocols include the use of the Digest authentication schema, the use of the Secure Socket Layer protocol (SSL) and the creation of the Secure Hypertext Transfer Protocol (S-HTTP). The first two focus on creating a trusted connection, the last on adding functionality to the basic HTTP protocol. Currently, these protocols should provide enough security for the implementation of a prototype.

5.8 Java applets

As said, a Java applet is a small Java program, running in a Web-browser. Basically, platform-independence is reached by turning Java source into so called ByteCode, which in turn needs a run time environment turning the ByteCode into machine-specific code. The ByteCode loaded into a Web-browser is what is generally called the Java applet, the browser provides the necessary run time environment. Figure 5-5 presents an overview of the Java structure found in a general Web environment. Java source files are compiled into Java class files, which contain platform independent ByteCode. These class files are downloaded by the browser, which contains the Java Platform, consisting of the Java API class files and an implementation of the Java Virtual Machine (Venners, 1997). In general, the combination of the Class Loader, the ByteCode verifier and the Execution engine is considered to be the Java

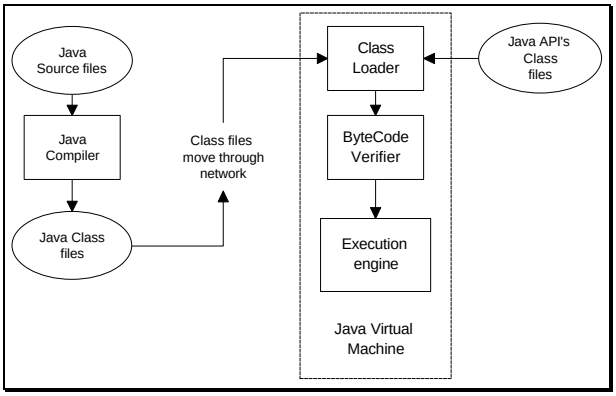


Figure 5-5 - Java environment on the Web

Virtual Machine (Venners, 1997). The Class Loader defines how Java classes are loaded over the network, while the ByteCode Verifier ensures that the code is valid Java Virtual Machine code and does not violate Java restrictions (Flanagan, 1996).
The Execution engine executes the ByteCodes. If done in software, there are basically two ways in which this engine is implemented: interpreting the bytecodes one at a time and using a just-in-time compiler where bytecodes are compiled to native machine code the first time the method is invoked.

5.8.1 Applet Security

Next to using the Class Loader and the ByteCode verifier, the SecurityManager class can be used to check whether requested operations are permitted (Flanagan, 1996). Web browsers create their own Security Manager and Class Loader to implement their security policies. The complete package of security policies implemented in Netscape Navigator 3.0 result in a number of restrictions being placed on Java applets, of which the following are important for the prototype (Flanagan, 1996), (Harold, 1997):

- They cannot access the local file system;
- They cannot create a network connection to any computer other than the one from which the applet itself was loaded;
- Applets cannot listen for or accept network connections on any port of the local system;
- They are not allowed to obtain any information about the user or the user's machine as they are available in the system's properties;
- Applets may not define any system properties;
- They are not allowed to specify classes which control how Java handles networking;
- It is not possible for an applet to invoke any program on the local system.

5.8.2 What to expect of core Java?

One of the major tradeoffs in Java has been execution speed. Interpreting bytecodes is 10 to 30 times slower than native execution, and Just-in-time compiling can only be 7 to 10 times faster than interpreting (Venners, 1997). In a networked environment, the Java Virtual Machine may have to wait for class files to download across a network, adding delay to the execution of the applet.
Java handles garbage collection automatically, but it is not always predictable when garbage collecting will start, and how long it will take. According to Venners, this makes Java a questionable candidate for software requiring real-time response to events.
Another problem can be called the 'lowest-common-denominator problem': if a certain feature exists on one operating system but not on others, support for that feature might not be included in the API. On the other hand if most operating systems share a feature which is not present on some other, this feature might be included through the implementation of a similar feature on the operating systems not supporting it standardly. For instance, the Java Abstract Windowing Toolkit (AWT) attempts to provide a user interface which adopts to the native look of the involved platform. It does this by using the java.awt.peer interface package, which defines the methods which must be supported by the GUI components on a specific platform (Flanagan, 1996). The Java AWT GUI components are implemented on top of the native GUI components of such a platform. Due to this approach, the look of a Java applet may vary across platforms. Basically, a subset of the look is copied, but also the problems inherent to a specific GUI are inherited.
Finally, it is possible for an applet to consume system resources, slowing a computer or a network connection down considerably (Flanagan, 1996). So, depending on the computer one uses, a certain applet might or might not cause these problems. Creating an applet on one

machine does not automatically mean it will run properly on another, which of course degrades actual platform independence. On many occasions during the creation of the prototype, what seemed to work on one platform, did not on another. For this reason, programming with the Java AWT means often programming with merely a subset of the AWT, decreasing the possibilities of this package.
In this case, the platform independent nature of Java applets justified the time it took to create the prototype. However, the problems identified above were indeed encountered, resulting in a sub-optimal result. Next to these problems, embedding Applets in a Web browser presented additional problems with respect to communication between Applets and between an Applet and the browser.

5.8.3 Applet to applet communication

The applet package within java consists of one Applet class, and three interfaces called 'AppletContext, AppletStub and AudioClip'. The way an applet interacts with the context in which it runs, is primarily defined in the AppletContext interface. This interface defines the methods which allow an applet to find out what other applets are running at the same time, use the Web browser to download an audio or image file and give a command to the browser to download a Web document or change the status info. When an applet is included in an HTML-page, it can be given a specific name. Another applet in the same browser can then communicate with this applet, by using the 'getApplet(String name)' method of the AppletContext interface. The following two pieces of code exemplify this.

```
import java.applet.*;

public class test extends Applet
{
  public String CurrentUser()
  {
    return "Jeroen";
  }
}
```

```
import java.applet.*;

public class test2 extends Applet
{
  public void init()
  {
    AppletContext AC = this.getAppletContext();
    test NA = (test) AC.getApplet("TestApplet");
    System.out.println(NA.CurrentUser());
  }
}
```

Here, the 'test' applet extends the Applet class by defining a method 'CurrentUser', which return a string with the current user. Applet test2 creates a reference NA to the applet which is named 'TestApplet' in the HTML-document, and calls the CurrentUser method. Finally, test2 writes the username to the standard output. This is the basic way in which applets in the same environment can communicate. However, the context in which an applet runs is confined to the frame in which it runs. Inter-frame communication between applets is therefore not possible. This proved to be a problem, since one of the tools created for the prototype was a 'guide', running in a separate frame. On one occasion, it was considered to be useful to let this guide applet communicate with the applet for the practical environment. The JavaScript ability to let different components within a browser communicate was useful here. The JavaScript 'LiveConnect' method is based on prior knowledge about the name of the applet, and the name of the frame in which it runs, thus reducing possible unwanted situations. For this, Netscape has added a number of classes to the standard Java API, making it possible to use JavaScript commands directly from within Java programs.

5.8.4 Applet to browser communication

As said before, the AppletContext interface defines a number of ways in which an applet can communicate with the browser. An applet can however never send its output directly to the browser, since no methods exist for this action. The only way an applet can ever establish a situation where it generates output, and commands the browser to show this is by temporarily saving it on the server side. The steps needed to do this are explained in Figure 5-6. The applet

uses the POST method to send a block of data containing the Web page to CGI program A, via the server. The server starts the CGI program, which stores the block of data in a unique file, and returns the filename to the applet. The applet than sends the 'showDocument' command to the browser, containing the URL of CGI script B, with the filename as the QueryString. The browser issues the request to the server, which starts CGI program B to retrieve the data. This data is then returned to the browser, which will display it.
Again, JavaScript offers much more flexibility in showing Web pages. It is obvious that Java lacks important methods to control the Web browser.

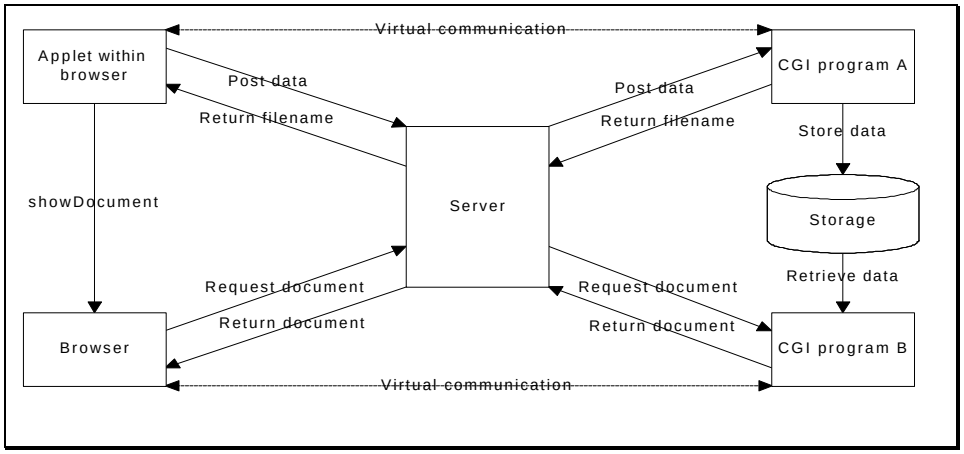


Figure 5-6 - Viewing an applet-generated document in a Web browser

5.9 Conclusions

This chapter is placed between the requirements planning part and the implementation part. As such, it can be seen as a technical risk-analysis of the environment, deciding on the possibilities and problems likely to occur in implementing the model presented in chapter 3. Due to the nature of the World Wide Web, the system is pushed into the area of a guided tour. Therefore, it can be predicted that the functionality presented to the student will be one of a sequential tour, involving a page-turning mechanism such as is warned for in chapter 3. But, within the limits of such an approach, it is likely that the Common Gateway Interface provides enough possibilities to function as the intermediary between the pedagogical application and the presentation components.
Adding more functionality to such a system, rapidly asks too much of a standard Web browser. The combination of HTML and JavaScript provides enough functionality at the client side to handle most requirements found at the user interface component of the model in chapter 3. However, where extending client/server communication is concerned, using either plugins or Java Applets can become a necessity. The platform-dependent nature of plugins make them unsuited, but the restrictions and limitations of Java Applets with respect to network, Applet to Applet and Applet to browser communication will likely result in sub-optimal behaviour of the system.
With respect to moving from risk-analysis to prototyping, existing Distance Learning systems on the World Wide Web have been looked at. According to (Minoli, 1996), Distance Learning solutions can be divided into Video/Audio based systems and Data-oriented systems. Currently, Video/Audio based system can primarily be found in Intranets, due to several problems described in the next chapter. Chapter 6 also describes existing World Wide Web based systems which can therefore all be classified as being data-oriented.

6. Distance Learning systems on the Internet

6.1 Introduction

In (Minoli, 1996), two types of distance learning technology are identified: audio/video based solutions and data-oriented solutions. Currently, most of the existing systems can be placed in the second category, since bandwidth problems on the Internet have restricted use of the first type of technology to Local Area Networks. However, current research into using more complex type of multimedia applications on the Internet can be traced down to two kind of developments: moving from a unicast network to a multicast network to reduce the load on the Internet as a whole and advances in streaming technology reducing the need for high bandwidth equipment at the user side. The first development has been extensively used in what is called the 'Mbone'. With respect to the second development, a company called 'RealNetworks' presents what seems to be an industry standard in real-time streaming multimedia delivery on the Internet. Due to these developments, data-oriented solutions are slowly adding simple forms of audio and video to their systems. The next part therefore focuses on the applicability of real-time multimedia for the Distance Learning system described in this thesis, followed by a description of a number of more data-oriented systems found on the World Wide Web.

6.2 Real-time delivery of multimedia on the World Wide Web

6.2.1 Streaming media

The standard TCP/IP protocols were originally devised for reliable transmission of data, with minimal constraints placed on experienced delay. However, multimedia is much more concerned with a real-time flow of audio and video, not needing the retransmission routines such as are used by TCP to provide a reliable end-to-end connection. In the case of video, delay due to retransmissions is often worse than the drop of one or more frames.

The Real-time Streaming Protocol (RTSP) is an application-level protocol aimed at providing a robust protocol for streaming multimedia. The current state-of-the-art with respect to streaming media on the Internet is presented by a system called RealMedia, which is therefore looked into more detail. This system requires the use of a plug-in. Next to presenting audio and video, the RealServer can notify the player when certain 'events' should occur. An event is basically a command the RealPlayer sends to the browser to retrieve a specific Web page. The standard delivery mechanism is based on UDP. When UDP is blocked by firewalls, the connection is established using the TCP protocol. The HTTP protocol is used in situations where a firewall filters out everything except Web access based on HTTP.

6.2.2 Adapting the presentation to available bandwidth

A large number of compression methods and combinations of audio and video can be used to create RealMedia files. This fact, combined with what is called 'bandwidth negotiation', should provide the needed flexibility in an environment where users connect to a server using many different speeds. However, RealNetwork's implementation of bandwidth negotiation is based on creating multiple RealAudio or RealVideo files for many different bandwidths. At connection time, the right file for the bandwidth the client has available is chosen. If a user does not have enough bandwidth available to play a certain file, buffering is used. So, if

someone tries to view a 100Kbps video using a 30Kbps connection and buffer download time is set to one minute, the video starts and stops at irregular intervals, making watching it very annoying. The best option therefore seems to be to make use of the rather simple form of bandwidth negotiation, and providing different qualities of movies to users with different connection speeds. In general, around 100Kbps seems to be needed to get a reasonable quality.

6.2.3 Unicasting and Multicasting

Originally, the Internet was set up to be a so called 'unicast' network, where data had to be send point-to-point. In this case, it means that the server has to set up a connection for each client requesting a video. This way, even using a dedicated E1 link can only support up to twenty simultaneous 100Kbps movies or thirty-six simultaneous 56Kbps movies.
Basically, there are two ways to overcome bandwidth problems: increasing the amount of bandwidth, or making better use of the existing amount of bandwidth. To use the existing bandwidth more efficiently, compression can be used, or the amount of redundant data-streams can be reduced. RealMedia offers different levels of compression, to reduce the amount of needed bandwidth (of course reducing the quality of video and/or sound). To overcome problems with redundant data-streams, Internet Multicasting (or IP Multicasting) has been invented.

The basic idea behind multicasting is that any piece of data only has to pass a router once on its way to multiple recipients. This way, redundant data-streams are avoided. However, where broadcasting typically involves sending a data-stream to all connected clients, multicasting sends a data-stream only to the clients requesting it. For example, this would result in a situation as is depicted in Figure 6-1. Here, six students watch movies at 400 Kbps, 100 Kbps and 58 Kbps.
This picture shows that there are only three data-streams from the server to backbone II, with a total bandwidth requirement of 556Kbps. No matter how many people watch those three video's at the same time, bandwidth requirements will stay the same at all connection points. Currently, the unicast model of the Internet is moving towards full support for multicasting by building multicasting into routers. In the past, special routers were created which transmitted multicast-packets wrapped inside unicast-packets to enable them to be transported over the unicast Internet. The global connection of such special routers was called the 'Mbone', which has been the largest effort to create a virtual multicast network on top of the Internet to support online multimedia distribution (see for example (Savetz, 1996) or (Kumar, 1996)).

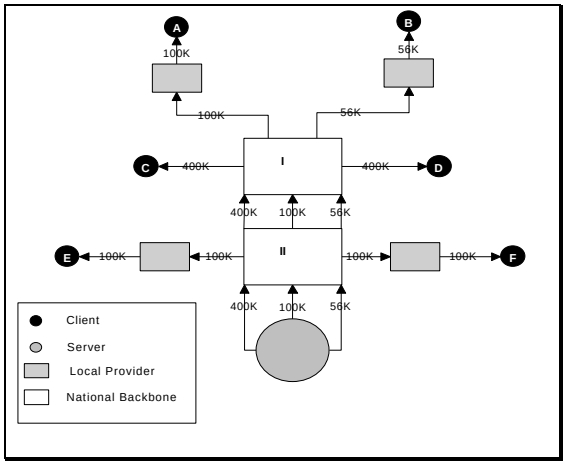


Figure 6-1 - Data-streams using IP Multicasting

6.2.4 Time-independence

In a situation based on time-independence, multicasting is of no use to transmit multimedia from a single source to a large number of destinations requesting different parts of the information. For example, if every 5 minutes a user starts some 100Kbps video available on the server, and the video is 45 minutes in length, the bandwidth requirements would never exceed 900 Kbps since after 45 minutes, for each person starting the video an other is done. At this

point, a T1 line would be in use for almost 60 percent, just so that 9 people can watch a 100Kbps video concurrently, but asynchronously. If all the server has to do is show that particular movie, or maybe more movies which use less bandwidth, some sort of primitive video-on-demand system would be sufficient. In this case, the system would have to follow some simple rules: start multicasting the video every five minutes unless there is no one to watch it. However, this is a sad compromise between using a typical synchronous technology such as multicasting, and an asynchronous system such as a time- and place independent educational environment. The idea of a single source providing the same information to a large number of end-users should be abandoned to make better use of multicasting technologies in these situations.

6.2.5 Distributing multimedia on the Internet

The approach taken with a proxy-model is based on 'pull', and requests to the Web-server hosting the original document will still be made asynchronously by many proxies. But in a closed system such as a Distance Learning environment, a push model might be appropriate. Each server would than act as a 'dynamic mirror' of the original site, and the multicast model can be used to make transmission as efficient as possible. It turned out that such a model is indeed in use in the United States to broadcast RealAudio and RealVideo on the Internet. This initiative is called RBN (Real Broadcast Network) and its primary focus is on providing live multimedia streams to a large audience. It is however also applicable to distribute RealAudio and RealVideo to 'mirror' servers which can then store the files locally.

A reasonable combination of sound and video requires at least a bandwidth of around 100Kbps, slowly becoming available to users. When we thus look at the situation at Southampton University, with an outgoing bandwidth of 10Mbps, in this phase no special arrangements seem to be necessary. Southampton University could not provide information about current usage of their outgoing line, but from information found at the website of JANET, it can be concluded that the outgoing link between Southampton University and JANET is around 25 percent occupied, leaving approximately 7.5Mbps for additional events. Using the full bandwidth available makes it possible to show around 75 100Kbps videos at the same time. However, the rather poor quality of such videos make them questionable for educational purposes, sheets for instance, can not be clearly shown this way. On the other hand, since sheets are rather static and do not change for a certain period of time, events can (and are) used to combine video with still images. Therefore, at the moment HPCnet could start using this form of video. However, future bandwidth developments at the student's side make it possible to use videos of higher quality. At that point, using a more distributed distance learning system will provide benefits. Such a system is described in appendix A.

6.3 Existing Web-based Distance Learning systems

To establish insight into current practice with respect to online course on the World Wide Web, I have looked for learning environment on WWW, which integrate several of the functionalities named in the previous chapters. The goal was to find environments which were focused completely on Distance Learning, as well as environments focused on supporting existing courses. After selection, the resulting systems provided insights into different factors:

- Using a convincing example to 'bridge the gap', or the move from the second layer into the first layer;
- Creating an environment supporting courses, and which can therefore be partly placed within the first layer of Kirschner's typology (chapter 1);

- Creating an environment providing complete Distance Learning, where communication between participants is only possible through the use of ICT. Such systems focus completely on the first layer.

The next part describes two systems falling inside the first and second category, the remaining parts focus completely on the last category of systems.

6.4 Providing communication and information

6.4.1 PDP-site

A large number of academic websites provide students with information about the courses they can follow, projects they can join, persons they can contact, etc. The pages of the faculty containing information about the department of Computer Science (WINS) at the University of Amsterdam is an example of such a site. This site is however in no way integrated with any course a student follows, and is an additional (and optional) extra.

A system which is currently being developed at Nijenrode University aims at providing extensive support for a part-time program, and is called the 'PDP-site'. It was initiated by the local Computer Centre, and in the past year it has served well as a convincing example of how ICT can support education. Nijenrode has in the past started a number of initiatives to use ICT in courses, such as a video-conference with a professor in the United States. The PDP-site is however the initiative which makes use of WWW-technology. The functionality provided by this can be summarized as follows:

- Each participant is known to the system and other participants through a profile, which can be changed by the participant. This way, the site can be adjusted to a persons wishes (within certain limitations);
- The site offers access to complete information about the courses which can be followed. For instance, news and announcements are available for each course, the schedule is available and course outlines, handouts and practice exams are put online;
- Discussions can take place on the global ('campus') level, or on the level of a course. This can take place in an offline as well as an online fashion;
- Each instructor can be reached through email, and information about an instructor can be found in his or her profile;
- Courses can be evaluated electronically;
- Some extra 'campus' functionality is offered through the possibility to view which students are online, and by viewing an automatically generated birthday calendar;
- The site is linked to an initiative taken by the Nijenrode Library, which offers an online catalogue as well as an extensive website.

Currently, the possibilities to offer complete courses through WWW are being looked into. As an example of this, the PDP-site offers a demo of an online course using RealMedia technology described before. Research is extended towards finding possibilities to allow students to work together on the Web.

Viewed in terms of chapter 1, this site has been created with as a primary focus providing information and communication. It soon proved to be very useful as a convincing example to 'bridge the gap' between innovators and the early majority of users. Although it did not offer any structured online courses, it is being perceived by students as a very good initiative, and instructors are starting to use it extensively. From interviews with mr. Kroodsma and drs. Groenenboom, it can be concluded that with respect to online courses Nijenrode is moving towards phase 4 of the lifecycle model of chapter 1. The importance of this example is that by

moving from the outer-layers to the inner-layers, an educational institution can bridge the 'gap', as is also said in (Kirschner, 1995).

6.4.2 Web Course in a Box (WCB)

An initiative similar to the PDP-site, but focused on providing an environment for a specific course instead of a whole program, is the Web Course in a Box system. It is aimed at providing tools for what is called 'Web-enhanced instruction'. Features present in this system are:

- Automatic generation of course/faculty/student home pages;
- Create online syllabus, schedule; manage student access to pages;
- Threaded discussion forums with file attachments/archiving;
- Upload content from instructor's PC / manage links to all pages;
- Use drill & practice quizzes / create linked lesson pages.

It offers six main menu's, namely 'Class Info', 'Announcements', 'Schedule', 'Students', 'Learning Links' (containing a Discussion facility and class-related Web-links) and 'Utilities'. It differs from the PDP-site initiative in its aim at using the Web as a part of a class, instead of as a supporting facility. Basically, although the resulting functionality is similar, the idea behind WCB is more aimed at the first layer.

6.5 Complete Distance Learning systems

To find out what the current state-of-the-art of Web-based education is, a number of integrated Web-education systems were compared with the system model described in chapter 3. This served two goals: first to see which parts of the model were and were not present in current systems, secondly to find out whether existing systems have functionality not present in the model. For this purpose, the following six systems were reviewed:

- DigitalThink, found at http://www.digitalthink.com/
- WebCT, found at http://homebrew.cs.ubc.ca/webct/
- TopClass, at http://www.wbtsystems.com/
- CyberProf, http://cyber.ccsr.uiuc.edu/cyberprof/
- LearningSpace, http://institute.lotus.com/LSpace/regapp.nsf
- and NIIT Netvarsity, at http://www.niitnetvarsity.com/

All offered demo-courses and all but one (NIIT NetVarsity) offered extensive documentation. LearningSpace differs from the other system because it aims at a so called 'Learning Team Centered' approach (see also chapter 2). However, this difference is mainly based on the philosophy behind the system, most LearningSpace components were also found in the other five systems.

As said, the systems were looked at with the system model described in chapter 3 in mind. The focus is therefore on:

- Subject Matter: contents of the pedagogical profile;
- Pedagogic component: structuring and style of presenting information, analysis of test-results;
- Student model component: characteristics of a student, such as results of tests and assessments and study-progress;

- User Interface component: user control, presentation and input analysis;
- Social Interaction.

6.6 Subject matter

6.6.1 Lecture contents

All systems provide online lecture notes, using multimedia such as graphical information and movies, which are used to describe complex situations. WebCT also offers a searchable image archive, to which an instructor can upload images to be included in the course. Images can be annotated, and students can search for images based on annotation, date of addition, image title, image creator, etc.
Other functionalities are quite similar to what is commonly used in books. Most systems offer the possibility to automatically create an index based on the contents of a course. Index entries can point to the appropriate pages. If such an index does not contain the term a student is looking for, some systems offer the possibility to use a search engine to search the complete course content.
Another possibility offered by these systems is the availability of a glossary of terms. WebCT even offers the possibility to add links from lecture notes to this glossary (under the control of the instructor).
The need for annotations as described in chapter 2 can be found in all systems, allowing students to add notes to the pages.
Links to external resources include links to an online library, an online bookstore and to appropriate pages available on the rest of the World Wide Web.

The expected parts are all available, and the ‘book metaphor’ adds several functionalities to the systems. Links to external resources are new, but not surprising.

6.6.2 Interaction between participants

The interaction possibilities found in the systems hardly go beyond the ones which have been available on the Internet for some time. Participants can communicate using a bulletin board system, a real-time chat facility or email messages. However, WebCT also offers students the possibility to show their results to their peers and instructors, which can then comment on those results. Here, the student creates Web based material, which can be included in the system. Some systems allow an instructor to keep logs of discussions, to gain better insights into the way students discuss subject matter.

In general, all systems offer functionalities already available on the Internet. The integration of these facilities to be used in an educational environment adds both ease of use and structure in discussions to such systems. Discussions are therefore focused on a central theme, based on the contents of the course.
Real-time chat facilities for pedagogical use are mostly available to talk to instructors online. However, many systems provide such chat systems as an extra, primarily for social interaction.

Most systems offer Frequently Asked Question (FAQ) lists, to lessen the need for an expert. For instance NIIT-varsity provides a facility called ‘Raise Your Hand’, which includes a FAQ database which is kept regularly updated, the possibility to ask an expert a question and newsgroups where fellow learners answer each other's questions. They say that historically, it

has been seen that more doubts are clarified with the help of peers than from the instructor. Finally students can mail a suggestion on the courses they follow.

6.6.3 Assignments and testing

All systems provide a way to make assignments online, and submit the results. Some systems added optional hints to the online assignments, of which an instructor using CyberProf states that the availability of such hints can answer many students' questions while they're completing the assignment. In his experience, it reduces the need for online availability of an instructor or assistant.
CyberProf also offers students the possibility to use a drawing tool to exemplify their answers. This sketch is then automatically interpreted by the server, returning comments on the graph. Assignments are logically placed in a 'lab', which could also provide room for simulations. However, extensive simulations are hardly used in the demo-courses I have followed.

All systems provide extensive support for testing, where the server chooses problems based on a random number generator. Multiple-choice answers are then also chosen at random. Online quizzes can be made available at a predetermined day, and some systems offer extra help by using audio, instead of plain text.

6.7 Pedagogic component

6.7.1 Structuring and style of presentation

All systems examined had a similar approach towards structuring information. Combined with the PDP-site and the WCB systems, the resulting structure for a course on the Web looks as depicted in Figure 6-2.
In this figure, the naming conventions are based on Webster's Dictionary:

- A study: something to be studied; a branch of knowledge. A study is usually broken up into several blocks consisting of courses;
- A course: a definite period of instruction and study in a certain subject, it is usually structured according to the model by Dousma (chapter 1);
- A module is a unit of coherent lessons. A module usually also obliges to the structure of a course described by Dousma, although testing is sometimes used for diagnostic use, instead of for selective use;
- A lesson is the smallest piece of education assigned by an instructor to be done at one time. It can consist of several Web-pages, combining one or more of the seven components identified by Collis (chapter 1).

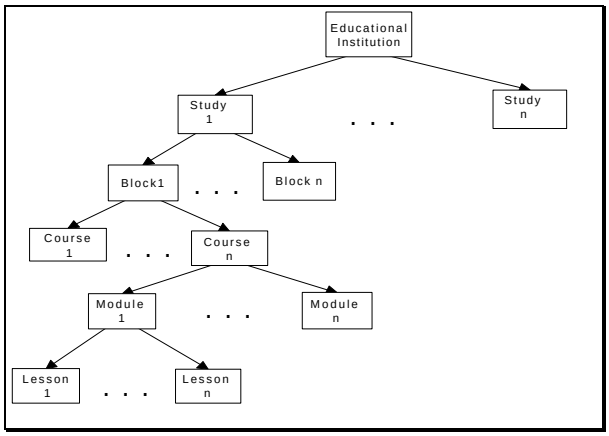


Figure 6-2 - The structure of an online course

All systems provide the user with a navigational toolbar, and a way to see the structure of the course. Navigation is mostly not much more than a 'previous' and 'next' button. In Figure 6-3, the so called DigitalThink 'Syllabus' can be seen. This presents the logical structure of the current course with the arrow indicating the current position of the student. Other systems provide similar overviews, none provided an overview more graphical than this one. The different ways in which systems have implemented the navigational part of the pedagogic component are all similar in their approach. All provide some sort of 'guide', which moves the student through the pre-defined structure of the course. To give the student somewhat more control, the systems provide them with an overview of the structure of a course, allowing the students to jump directly to their subject of interest.

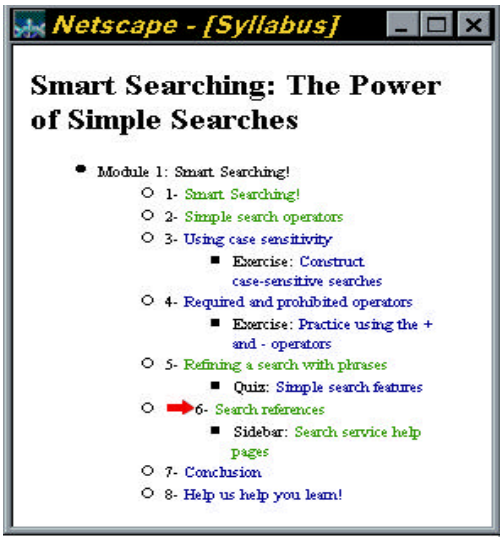


Figure 6-3 - The DigitalThink Syllabus

6.7.2 Analyzing test-results

Automatic analyzing of test-results can be found in most systems. Students then receive immediate grading and obtain detailed information on how or why their solutions are incorrect. They can compare their results with the minimum, maximum and mean of the results of their peers. All systems record the time it takes for a student to solve a problem, and this can be compared to the expected time. Test results can be used by an instructor to adapt a course to a student's needs by for instance assigning extra material.

The most sophisticated testing system was found in TopClass. Using the 'auto-testing' facility, the server can take actions based on how an individual student performs. It can do such things as automatically assigning additional course material or notifying an instructor if the student scores above or below a predefined threshold.

Most systems provide extensive support for testing students' abilities. All systems provided feedback on the reasons why answers are wrong or correct, and how an individual student is doing compared to his or her peers. Instructors were able to act on a student's results, but only in TopClass could the system act on results.

6.8 Student model

The student model served two purposes: allowing the system to keep track of the student's position within a course, and monitoring the progress of a student with respect to understanding the subject matter. Each time a student re-enters a course, he or she starts at the position where he or she left the last time. WebCT offers a 'Grade Tool', where a student can view his or her marks as entered by the designer, along with any comments made by the instructor. Each mark ever given is kept in this component, available to both student and instructor.

6.9 User Interface

Since all systems are based on WWW technology, the user interface consisted of a Web-browser. On many occasions, several plug-ins had to be added to a standard browser, and DigitalThink therefore offers direct access to their software store.

Input is normally checked on errors, although this is often done at the server-end. This is because a Web-browser is basically 'dumb', and needs things as plug-ins, Java-applets and Java-Script to add 'intelligence' to it. More or less unexpected, the use of adaptive toolbars functioning as a guide was found in almost all systems. The contents of such a toolbar adjusted itself to the extra options offered with respect to the page currently viewed (such as taking a self-test, making annotations, etc.). In Figure 6-4, the toolbar found at the WebCT system is presented.

Figure 6-4 - Adaptive toolbar, found at WebCT

6.10 Social interaction

Here too do all systems offer functionalities already available on the Internet. The facilities used for social interaction are similar to the ones used as part of the pedagogical profile. Although here there is no logging, and discussions are not focused on a central theme. Next to the standard online and offline discussion utilities, DigitalThink was the only system offering a utility to send short messages to online classmates. It served basically as a way to inform people you were there, and possibly to invite them to the online chat area.
Most systems place tools for social interaction separated from the pedagogical components. WebCT for instance has placed the online chat utility on their 'Tool Page', outside of the real course.

6.11 Conclusions

From the descriptions above, it can be concluded that most of the theoretical functionality is available in current systems. The approach towards structuring information is based on a 'page-turn' mechanism, which is due to the combination of hypertext and the sequential nature of courses.
However, what is missing in most systems is autonomy on the system's part, in terms of automatically adjusting a course to the needs of a specific student. In most systems, this is done by the instructor and sometimes this is not done at all.
Another thing missing is the integration of all parts having to do with getting expert help. Only NIIT-varsity offers an attempt at providing such an integrated environment.
One part which was not clearly identified in the model presented in chapter 4, was the use of an adjustable toolbar, or 'guide'. This seems to have become a standard approach towards guiding students through the online learning environment.
None of the demo-courses included a clearly identifiable lab, where assignments could be done resembling the functionality of a real lab.

With respect to the system described in this thesis, these findings have the following consequences:

- The 'book-metaphor' is a common approach towards creating an environment as described in chapter 3. One of the major requirements here is the use of an adaptive toolbar guiding

students through the environment. The combination of using a Java Applet and a remote server to provide such a guide is described in chapter 7;
- Although theoretically described before, adaptive learning environments are not yet generally used. To be able to build the final system, more research will have to be done in this area;
- Focusing on the 'integration' requirement, the social interaction environments found do not satisfy this need. Integrating such an environment within the complete system should get more attention;
- The use of audio and video is limited by the bandwidth at the student's side, it's availability is rather primitive. However, advances in bandwidth technology might soon provide an environment in which such multimedia can be used without large problems;
- For this thesis, the creation of an environment for practical programming assignments is of major importance. The requirements for such an environment have been researched by implementing a prototype described in chapter 7.

The final chapter of this thesis describes the implementation of part of the prototype.

7. Implementing the prototype

7.1 Introduction

In the previous chapters, the requirements for a distance learning system have been described. In this chapter, a prototype of such a system based on World Wide Web technology is described in more detail. To complete the description of a Distance Learning System this chapter focuses on three components: the user guide, the discussion facility and a virtual computer lab.

7.2 Current situation

The case example used for this thesis is based on the CERN School of Computing course about Parallel- and Distributed Computing, given at Egmond aan Zee in the Netherlands from 8 september till 21 september 1996. This course consisted of a set of lectures and a tutorial on the practical aspects of parallel and distributed computing. Using this tutorial, students needed to develop two parallel programs which were then executed on a cluster of workstations present at the school, and on a massively parallel computer, a 512 node Parsytec Gcel which is located in Amsterdam. For this thesis, an environment for executing code on the parallel machine was developed, connecting a Web browser with a machine named 'hydra' located at SARA which functions as a front-end to the Parsytec Gcel.
The course lasted two weeks, and aimed at providing students with a good idea of what parallel and distributed computing is about. Since it was based at a complete lecture called 'Introduction to Parallel Computing' taught at the University of Amsterdam, findings from this case example can be used as a basis for moving the complete course to a distance learning environment. For this prototype, information found in the tutorial was primarily used.

7.2.1 Requirements for the course

The tutorial consists of five chapters: an introduction, a short review of terminology, presentation of theory behind simulating a plucked string, an introduction to PVM programming and finally the assignments. Practical help can be found in three appendices about PVM and the plucked string problem. A distance learning environment should therefore at least provide a guided tour through the subject matter and the availability of an environment for these particular practical assignments. Both for discussing the subject matter as well as asking questions about the assignments, the environment should provide the possibility for a student to ask questions to experts, or turn to their peers for help. In the following part, principles behind a guided tour and interaction between participants is described in more detail.

7.3 Providing a guided tour mechanism

To support the type of structure presented in the previous chapters, a system should include some sort of 'guide', which communicates with the server about which part of the course should next be visited by the student, or which can present an overview of the course, pointing out where the student is at the moment. For this, knowledge about both the structure of a course and a student's current position is needed. In such a schema, the guide would only have to command the browser to get the next page of the current module or show an overview. This is supported by the Applet-browser interaction. This way, the Applet commands the browser to get a particular page, the browser requests this page from the server and the server adjusts

the student's current position. A rather static approach would be to have the Applet download the complete course-structure at startup, and have it command the browser to request predefined pages. Another way would be to create a CGI-program called something like 'GetNextLesson' at the server-side, which both updates the student's current position as well as returns the next lesson to the browser, depicted in the left part of Figure 7-1.

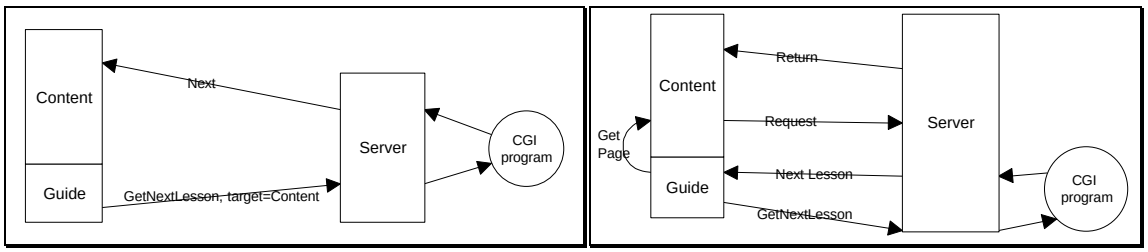


Figure 7-1 - Simple and complex interaction between guide and server

This approach is rather easy to implement, both at the server as well as the client side. The client side doesn't need any intelligence, since the call for a next lesson, a previous lesson or an overview is always the same for the client. It is the server which handles the requests and needs the intelligence. However, if a guide enables a user to do more than just browse through pages in a structural manner, it needs its own intelligence. Many distance learning environments on the Web also put buttons for questioning, discussions and quizzing in a guide. Depending on the extra functionality provided by the active lesson, the guide adds or removes buttons from its interface. In such a situation, an approach similar to the one above can be taken. Now, it is the guide which communicates with the server, instead of the browser. The guide still sends commands like 'GetNextLesson' to the server, but now the server returns the necessary meta-information for the lesson to the guide. Included in this meta-information is the location of the page requested, which is used by the guide to command the browser to download that specific page, this is depicted at the right side of Figure 7-1. The remainder of the meta-information is used to adjust the guide to the needs presented by the current lesson. Next to this, the guide can contain such information as the user's identity, making this available for other applets used within the course. It is the complex Java guide which has been used for the prototype of this project.

7.4 Following links outside of the guided tour

With using the World Wide Web and its inherited hyperlink structure, the problem of a student following links which will lead him or her outside of the domain of the guided tour arises. In such a situation, a student will get lost quickly, since after a number of subsequent steps outside the guided tour, a logical connection with the lesson will be lost. There are several solutions to this problem, differing in the amount of control the Distance Learning system has over the external pages. An approach letting the system keep almost full control is called 'Footsteps' (Nicol, 1995), and is based on redirecting every link in the requested page to a CGI script on the system, which will then handle the request for the student. The system itself is presented as a guided tour mechanism, deciding for a user which link is next. Whenever a link to a resource outside of the guided tour is followed, this request is handled by the system itself. The system will then make a request for the page, replacing every link in that page with a call to the system itself, while adding a guide to the page including a 'Return to tour' button. According to Nicol, the main disadvantages of this approach are that all objects are retrieved in a two-stage process. The Footsteps approach is a very good approach towards providing a

guided tour through external Web pages. However, if all pages within the guided tour are under the control of the system itself, a much simpler approach can be taken.
The approach taken in the prototype is based on moving the action taken when leaving the guided tour to the client's browser. Whenever an external link is followed, a new browser window is started. If the student wishes to return to the guided tour, he or she simply has to close the new window. However, without a proper warning, a user might experience disorientation because of the fact that he or she does not know a new window has. Therefore, such an approach needs a mechanism which both informs the user that he or she follows a path which will lead them away from the actual guided tour, and that this will happen in a new browser window. An intuitive approach can be taken by using JavaScript. Here, when an external link is chosen, a small window pops up informing the user of the consequence of his or her choice adding a 'Cancel' and 'OK' button. This results in a window such as in Figure 7-2.

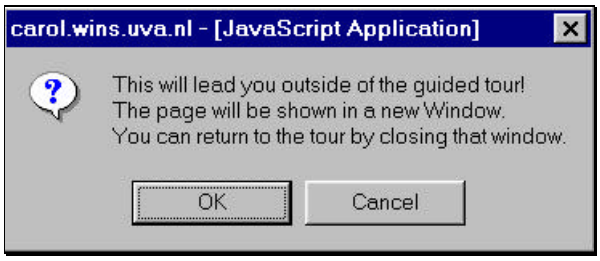


Figure 7-2 - Window to inform user of diversion from guided path

Consequence of this approach is that any page included in the guided tour must be checked beforehand on the presence of external links, which should then be redirected to the Java script. In this respect, it is similar to the footsteps approach, but does not place any constraints on external pages.

7.5 Social interaction within the environment

The creation of an environment for social interaction proved to be of a very complex nature. The combination of the Answer Garden and the Knowledge Tree approach was identified as being useable in the environment as described in previous chapters. Although I have implemented a very rough type of shell for such a communication environment, a useable environment could not be build within this project. One of the conclusions are that an SQL database can be used to store both the structure and the contents of the question and answer environment. Such a database should then consist of a table containing all concepts and any information needed about the concepts, a table containing all links between those concepts consisting of a parent-child pair combined with the type of the link, and finally three tables containing all questions, answers and discussions belonging to a specific concept.
Implementation of the user-interface proved to be not too complex. Basically, two approaches were taken: building a Java-applet and using a plain Web-browser. The intermediary between the user-interface and the system is formed by a standard web-server, therefore the prototype relies on using CGI. This way, standard procedures for establishing a user's identity can be used, and functionality already build in a web-browser and web-server did not have to be re-invented. The interface basically provided a hyper-linked view of the database structure. A JAVA-applet can use the provided output to perform actions, a standard Web-browser merely displays that output on the screen. Therefore, much of the intelligence of the user interface lies at the CGI-script when a standard web-browser is used, and at the JAVA-applet when this is used. Clearly, such a system can not work properly when a QA-Tracker is concerned, since this has to work independently of any user. This requires an extra system on the server side, tracking the status of all questions which has not been build. There seems to be no reason to not decide to include such an environment in the final system.

7.6 Basic requirements for a Virtual Computer Lab

The three requirements identified in the introduction have been translated to creating a virtual computer lab in the following way:

- Time independence: this basic requirement is already available in a normal practical assignment situation and is not changed for this implementation. It should basically be possible to have access to the system at any time, enabling a student to work on his or her source code and testing it out on the Parsytec Gcel;
- Place independence: this requirement has two consequences for this prototype. First, it brings with it the need to store files centrally, since a student must have access to his or her files from any place. Next, a student must be able to use different types of computers on different locations. Therefore, the virtual computer lab must be able to run on any platform used;
- Integration with the complete distance learning system: the virtual computer lab is an instance of the fifth component described by Collis and presented in chapter 1, and as such belong to the 'Concept presentations' component of the model presented in chapter 3. This means that for the system described in this thesis, the virtual computer lab should be 'seamlessly' integrated with the World Wide Web environment of the complete system.

The environment which comes closest to the 'integration' requirement is the use of Java Applets within a Web browser. This also provided the possibility to communicate with a centralized storage facility, needed to provide place independence. Finally, networking capabilities of Java Applets also provide the means for communication with the Parsytec Gcel. This chapter therefore provides a detailed description of the implementation of a prototype of a virtual computer lab based on the use of Java Applets within a Web browser, and the consequences this choice has on the functionality provided to the user.

7.7 Components of a practical assignment environment

In the current situation, the students are provided with the following components:

- a source code editor;
- a compiler;
- file i/o;
- a runtime environment located on the Parsytec Gcel;
- a graphical display of results (GNU plot);
- manual pages;
- expert help.

In building a distance learning system, all (graphical) output has to be displayed on the student's web browser, which is also responsible for all event-related interaction with the user. There are four identifiable global components, as depicted in Figure 7-3.

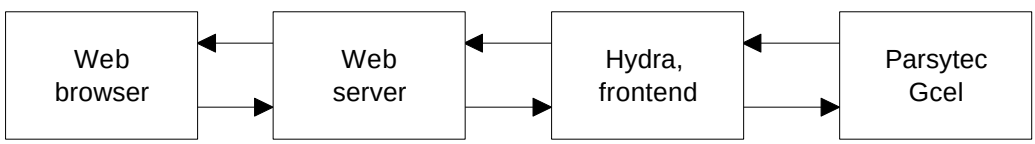


Figure 7-3 - Global components of the Virtual Computer Lab

Due to restrictions on Applet network connections described in chapter 6, a Web browser cannot communicate directly with the front-end machine, unless the Web server is installed there. Since this was not the case, the machine the Web server was installed on had to behave as an intermediary between the Web browser and Hydra. The way Hydra interacts with the Parsytec Gcel is of minor importance, since this does not differ from the normal situation. The implementation has therefore followed a simplified three-tier client/server paradigm, in which the following three components can be identified:

- A Java Applet client running at the student's computer within his or her Web browser. This Applet presents the user with all necessary UI transactions and a source code editor;
- A Perl server, located at Hydra. This server provides all compiler, plot and runtime services along with file i/o and manual pages;
- A tunnel written in Perl, located on the server from which the Applet originated. This ensures flexibility on the location of the Perl server.

As was said before, the World Wide Web provides services for an event-driven architecture, initiated by the client. It uses a stateless protocol, basically limiting communication to a request-response method. Execution of a program on a remote site creates the need for a different communication protocol. Client and server need to communicate continuously, for instance when the remote program needs input from the user. Because of this, the HTTP protocol has been abandoned, and the possibility of an applet using TCP directly was used. This also created the need to develop a protocol to support the application layer.
The tunnel between client and server currently is 'dumb', and does not use the information being transmitted. As such, it can be seen as part of the middleware enabling client and server to communicate. Since the goal of the Virtual Computer Lab is to provide a distributed application, middleware is placed between one part of the Application Logic component, and an other part of that same component. The consequence is that the Application Logic component is split in two, resulting in a need to decide on where to place the different actions, calculations and decisions of this component. Most of the interaction between client and server can be described by the RPC. However, above is described that there is need for Message and Queuing technology when client and server communicate to allow a user to interact with the runtime environment. Therefore, middleware aimed at both RPC and MQ has been developed. The complete environment standardly available to a user on the Hydra machine should thus be made accessible to a remote student. This chapter first focuses on the communication protocol used, after which all components are described in detail.

7.8 Description of the Client Applet

The Applet at the client side must provide a lot of the functionality normally found at the front-end machine. The prototype implemented for this thesis contains the following identifiable parts at the client side:

- A component enabling all communication between the client and the server;
- A login procedure;
- A source code editor, in this implementation this is merely an instantiation of the TextArea class found in the standard java.awt package;
- STDOUT and STDERR functionality;
- File I/O, containing loading and saving of files, along with the possibility to view remote directories;
- The front-end to a runtime environment;
- The front-end to a compiler;
- A graphical display, showing output provided by GNUPlot running at the server;
- The front-end to requesting and displaying standard manual pages;
- A link to the question/answer environment described in the previous chapter, to provide expert and peer help.

In this prototype, data send by the client is handled by the PrintStream class found in the standard Java io package. This class implements a number of methods for outputting textual representations of Java primitive data types, instead of their binary representations. Since client output consists of either a text file containing the source code or textual commands for the server to interpret, the PrintStream class presents the best choice. However, the data-stream outputted by the server requires both interpretation as well as actions from the client. For use within this prototype, four different input streams have been identified: StringBuffer, Line, FIFO and List, as presented in Figure 7-4.

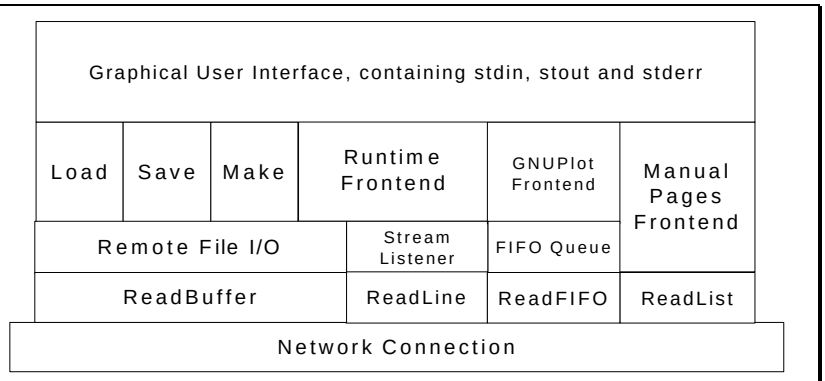


Figure 7-4 - Components using different server output streams

All of the components present output to the GUI presented to the user, and are presented on this GUI via a button. The reason behind these choices is described in the next parts.

7.8.1 Communication between client and server

To provide all the necessary communication at the client side, a 'Connection' class has been written in Java, consisting of the following methods:

- Connection: initializing the connection;
- TryConnection: opens a socket and connects and inputstream and an outputstream to this socket. If failure occurs here, the client can either not connect to the tunnel, or the tunnel cannot connect to the server. If successful, it calls the 'HandShake' method;
- HandShake: this function performs a handshake with the server based on a simple protocol:
    - Get Welcome message, including Server version;
    - Send username and Applet-identifier;
    - If access granted send password, else call CloseConnection;
    - If password is not accepted, call CloseConnection.
- Write: writes a string to the server;
- ReadBuffer: reads a buffer of strings from the server;
- ReadList: reads a vector list from the server, creating a randomly accessible list of strings;
- ReadFIFO: reads a string from the server, and puts this on a FIFO queue. This enables the client to perform actions on the data while retrieving it;
- ReadLine: read a string from the server;
- CloseConnection: closes the connection.

Each Remote Procedure Call first calls TryConnection, and when successful either writes to or reads from the socket. In the detailed description of the Client which follows next, it will become clear how each component makes use of this class.

7.9 File I/O

File I/O basically consists of requesting and retrieving a directory listing, retrieving a file from the server and storing a file on the server. Retrieval and storage of a file is based on providing the server with the path and filename of the file. In the prototype, this is always done through an instantiation of a class called 'DirInfo'. This specific class allows the client to communicate with the server to retrieve directory information, and provides an interface so the user can either choose an existing file or create a new one as depicted in Figure 7-5.

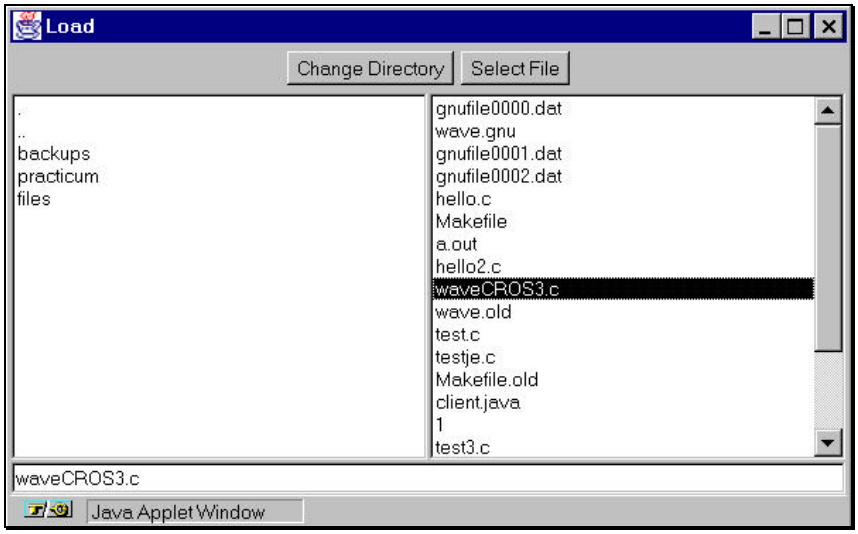


Figure 7-5 - User interface for retrieving directory information

When this is done, the path and filename are returned to the object instantiating the DirInfo class, and data is either written to or retrieved from a socket. Current directory information is stored in a static variable, making it available throughout the session. The server mainly reads directory information, outputs the content of a file to the client or creates a new file with content provided by the client as will be described later. The prototype uses a similar construction for compilation of code: after the code is stored on the server, DirInfo is instantiated to allow the user to supply the client with the name of the MakeFile and parameters to this MakeFile. Then, the client sends the right command to the server, waits for the results, and displays these results on the user's screen.
Implementing such an approach in Java presents some difficulties. First, the standard library of File I/O classes is useless in a situation where files are stored remotely. Next, the DirInfo class created for this purpose displays a new window and adds event-handlers to it, after which the main class exits, leaving the window and event-handlers active while returning control to the calling object. At that point, the user has not yet decided on any filename, therefore the calling object is not able to proceed. The solution chosen here is using an interface called 'FileInterface'. Each class which using the DirInfo class, should then implement this interface. Below, a piece of code from the prototype is presented, which is used to load a specific remote file. In this piece of code, it can be seen that the contents of the file are outputted to the user input window, while system messages are sent to a separate window. This piece of code is connected to both the user interface and the DirInfo class through the use of event handlers. When a user clicks the 'Load' button, this results first in an instantiation of the LoadClass, and secondly in an instantiation of the DirInfo class containing the previously instantiated LoadClass object. When the user has chosen a specific filename, the DirInfo event handler makes a call to the LoadClass.Action method which is an implementation of

FileInterface.Action. It is this method which is responsible for handling the loading of the file as can be seen in the code above.
This general approach is used in all components using remote File I/O.

```
interface FileInterface
{
	public void Action(String dir, String file);
}

class LoadClass implements FileInterface
{
	Connection con;
	EditorInterface inputarea;
	OutputInterface outputarea;

	public LoadClass(Connection con, EditorInterace inputarea, OutputInterface
outputarea)
	{
		this.con=con;
		this.inputarea=inputarea;
		this.outputarea=outputarea;
	}

	public void Action(String dir, String file)
	{
		String S;

		if (con.TryConnection())
		{
			con.Write("load");
			con.Write(dir);
			con.Write(file);
			con.ReadErrors();
			inputarea.setText(con.ReadBuffer());
			outputarea.appendText("Loaded file "+file+"\n");
			con.CloseConnection();
		}
	}
}
```

7.10 Runtime environment

The runtime environment uses MQ to communicate with the server. Again, using the DirInfo class, a user provides the client with the path and name of the executable. The client sends a message to the server, requesting the server to run the specific executable. The server then listens to both the client and the running process, and is an intermediary in their communication. The way this is presented to the user is displayed in Figure 7-6. The major problem here was that in a Unix environment, a process will only use non-buffering I/O if it is connected to a real terminal. In any other case, it will buffer its output, resulting in a form of dead-lock: the process sends its output to a buffer, and waits for input from the client. The client waits for output from the process, but does not receive it until the buffer is purged. Clearly, buffering i/o presents a problem here. From the above description two solutions arise:

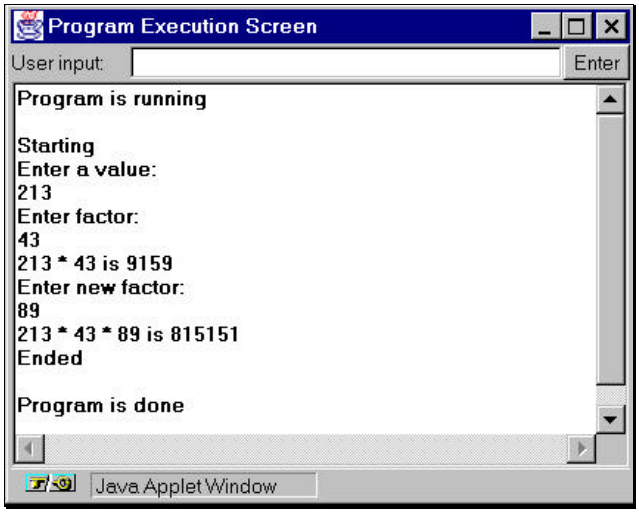


Figure 7-6 - User interacting with a program running on the server

- making sure the process does not use buffered i/o;
- emulating a terminal when communicating with the process.

Both options have been tried, and both work. The first solution is not preferable, since it requires either rewriting standard libraries or rewriting the program which is executed by the

user. However, it required not much more from the server than to connect stdin, stdout and sterr to the running process. The second solution in Perl relies on non-standard routines and requires the server to be more active in sending data to the process and capturing output from the process. Both approaches are described in somewhat more detail when the server is described. On the client side, only a small piece of coded is needed to enable interaction with a program running at the server side.

It can be described by three different steps:
1. Create Execute Window, containing a user input line and a program output area;
2. Create a thread listening to the server;
3. Create an event-handler redirecting user input to the server.

The thread runs synchronously with the Execute Window, and redirects all server output to that window. In this case, only textual information is send by the server, making this a feasible approach. When the connection with the server is closed, the thread outputs a message to the Execute Window and exits, the client only tries to send user input to the server as long as the thread is still alive.

7.11 Using GNUplot for a graphical display of results

Implementation of this component turned out to be rather problematic. First because a way had to be found to provide the client with the graphs plotted by GNUPlot, next because retrieving such graphs over a standard modem connection can take a while, it was preferred to plot the graphs on the user side while retrieving them. The first problem required the use of an intermediary language describing the graphs. The second problem required using a thread reading the input from the Connection class in a First-In-First-Out manner. Therefore, a graph description language based on lines of text was looked for. The solution has been to use a standard set available in GNUplot, and write an interpreter in Java able to translate the commands into Java graphical commands. A number of different terminal types can be chosen in GNUplot, resulting in different behaviour. When the ‘xlib’ type is chosen, GNUplot outputs a list of commands in string format, basically describing a graph in terms of text, points and lines. Tcl/Tk code using xlib commands was available and was provided to me by Robert Belleman. This code has served as the basis for the graphical Java interpreter. So, again, the server outputs a number of strings which are used at the client side to perform actions on the user interface. Basically, this description language has the following format: 1 byte for the command, 4 bytes for the x-axis, 4 bytes for the y-axis and a string. The commands used in the prototype are:
- G: clear window;
- M: move pencil to position (x,y);
- J: define anchor for text: 0 means place text left from current position, 1 means center text and 2 means align text right from current position;
- V: draw line from current position to (x,y), and make (x,y) current position;
- L: set line width;
- T: place text at current position, based on the anchor value set previously.

This approach does have its drawbacks. As can be seen, there is no indication of how large the output window should be. This is in general set by the user through commands send to GNUPlot, but are now unknown to the client. The approach used for the Client Applet was to wait until one complete graph was loaded, and then calculate the size of the canvas according

to the largest and smallest x and y values. This size was then divided by the size of the Applet window, and the result used to decide on the real length of the lines. This way, resizing the Applet window resulted in a redraw of the graph fitting the size of the window. On the user side, the display was as is shown in Figure 7-7.

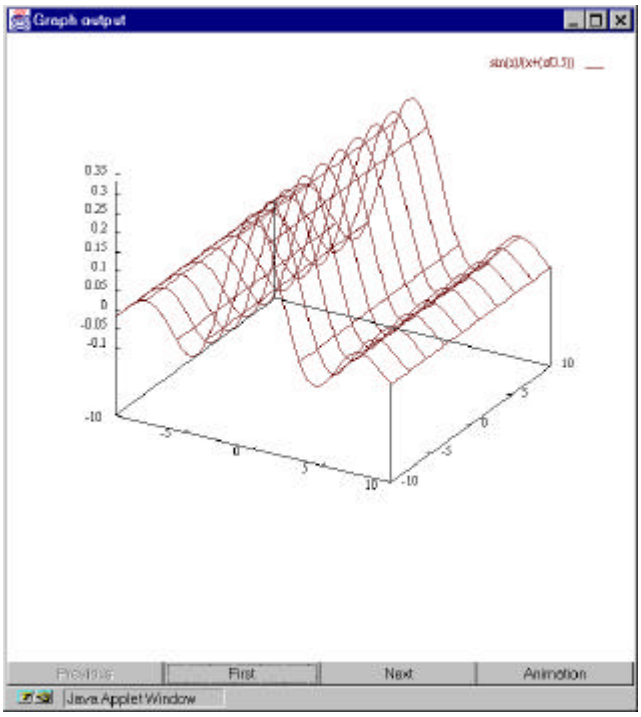


Figure 7-7 - Client output of GNUPlot data

In this situation, the choice was made to display the graphs while they were being received by an instantiation of the Connection class. Thus, the ReadFIFO method was used. This method connects an object which processes xlib strings to a FIFO queue onto which the ReadFIFO method pushes incoming strings outputted by the server.

The strings from the server are translated into an intermediate format until the end of a graph is reached. Then, the intermediate format is used to quickly draw the graph on the output window while the next graph is being received.
However, the use of multiple threads to allow for such actions used up a large amount of CPU time for a relatively long period of time, resulting in noticeable reduction of performance on a standard 133Mhz Pentium PC. Next to this, socket communication for large amounts of data seems to be slower than expected using a Java applet, increasing download time (using compression will be of some help here). Finally, problems were encountered when the Applet was used on different platforms. Observed behaviour included disappearing of the output window until all graphs were loaded, missing colors and difference in behaviour when the output window was being resized. After loading the graphs, a student can view any of the graphs using ‘Previous’ and ‘Next’ buttons. Animation of the graphs was done using code from the Animator class found in (Flanagan, 1996), which is based on starting a thread which displays a graph and then sleeps for a specified period of time. Overall, the approach works relatively well for this specific situation. This approach is however not comparable to for instance an approach such as the X-Windows system, where a complete graphical User Interface is distributed among a client and a server. However, such an approach is not feasible due to bandwidth restrictions as described in chapter 6.

7.12 Manual Pages

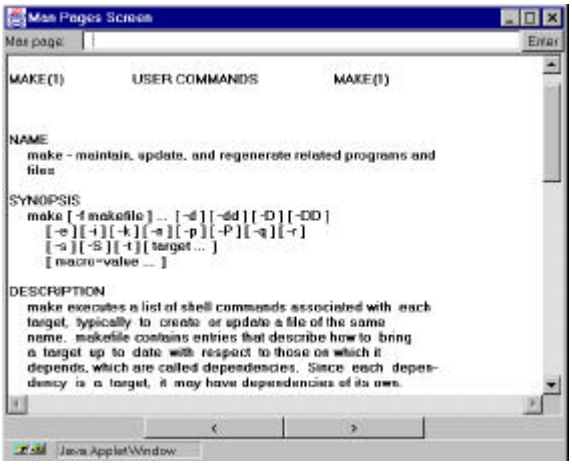


Figure 7-8 - Manual pages User interface

The availability of standard 'man' pages are useful when programming in a Unix environment. Such a 'manual' consists of (possibly) a large number of pages, in this case outputted by the server as one large document. Both because of the fact that on some platforms putting a large amount of text in a TextArea can cause problems and because such a large unformatted document is rather unreadable, the client formats the document into the right number of pages again. To be able to format incoming text, it was necessary to read a list of strings, instead of a block of text. The user interface basically consists of a line for user input, and an output window to present the contents of the manual pages as depicted in Figure 7-8. A simple 'previous' and 'forward' button enables a student to browse through the different pages.

7.13 Using a tunnel as an intermediary

The function of the tunnel in this prototype is rather simple: providing the 'glue' between the client and the server. It accepts a connection from the client, and then connects this client-socket to the remote server. It then reads client-output, redirecting this to the server and vice versa. Furthermore, it is a dumb application, not aware of the contents of the data it redirects. Therefore, it should present the following functionality: connection a client socket to a server socket, closing the client connection when the server connection is closed and closing the server connection when the client connection is closed;

For the client, the tunnel functions as a server, for the server, it functions as a client. This requires first of all the creation of a server-socket, connected to a port by the client. This socket is created at startup of the tunnel program. From then on, the functions of the tunnel can be described by the following steps:

- Listening for a connection request from a client;
- Accepting such a request, redirecting it to another socket;
- Opening a new socket to the server;
- Forking a process to redirect client output to the server;
- Letting this forked process fork a new process to redirect server output to the client;
- Return to listening for connections.

Standard Perl signal handling was used to provide communication between the two new processes. This allowed the creation of subroutines which are invoked when Perl catches a signal. The way this was implemented in the tunnel was by letting the subroutines abort the server process when the client connection was closed, and vice versa. Next to this, termination of the main program also resulted in aborting all running child processes.

7.14 Implementation of the server

The server is located on the front-end machine, in this case the 'Hydra' machine. It is also written in Perl. It implements a server socket to listen for connections, and executes code to run programs on the local machines dependent on the commands it gets from the client. Identification of a user is done at very basic level: a client supplies both the username and password in plain text format, and these are checked against information available at the server machine. The server provides the other end of the communication protocol described in the client section of this chapter. Functions provided by the server can be summed up as follows:

- Listening for connections on the server port;
- Accepting requests, redirecting them to another socket;
- Forking a process handling the request, returning the main process to listening for new connections;
- Identifying the student;
- Reading the command line;
- Executing the command, based on client input. Output is redirected to the client.

Most commands follow a basic approach, which is exemplified by the request of manpages on the next page. Here, NS represents the Network Socket connection with the client. After identifying that the client has requested a manual page, this information is outputted to the server monitoring screen. Then, the parameters to this request are read, which are validated to prevent the user from sending malicious commands to the remote machine. Before executing the command on the local system, STDERR is redirected to the client.

```
if ($line eq "manpage")
{
   &OutputConsole("manpage requested");
   $parameters = <NS>;                              # Read parameters
   if (!(&ValidateManParameters($parameters)))      # Validate parameters
   {
      &CommandFailure("Invalid man page requested");
      exit;
   }
   $manpage=&ExecuteCommand("man $mancommand");     # Translate query into call
   $manpage=~s/.\cH//g;                             # Make output client readable
   &OutputResults($manpage);                        # Send results to client
}

sub ExecuteCommand
{
   local($command)=@_;
   open(STDERR,">&NS");
   $output=`$command`;
   close(STDERR);
   return $output;
}

sub OutputResults
{
   local($output)=@_;
   print NS "\0stdout:\n";
   print NS "$output";
}
```

The output of the local command is captured in a program variable, which (if needed) is then transformed into a block of text readable by the client. The client is notified that the following piece of text consist of the output presented by the program, and is informed of the length of that specific piece of text. After this, the output is send to the client. It is not important whether either STDERR or STDOUT is empty, the client code handles this.
Closing the client connection is done in the main program. For other commands, such as the one requesting directory information, no direct system call is used. Instead (in this specific case) directory information is read, and information about each file or directory found is send to the client. This information includes the type of the entry (whether it is a file or directory

and whether the user has rights to write, read and execute the entry), the name and size of the entry and the time and date the entry was last modified.
The one command which could not be implemented this way is the routine executing a user specified program on the server machine. Since STDERR, STDOUT and STDIN are all in use concurrently, distinguishing STDERR from STDOUT output was difficult. Next to this, the earlier mentioned problem with buffered I/O needed to be solved. In the implemented prototype, both STDOUT and STDERR were connected to the client, resulting in a mixed output at the client side (since normally they are both connected to the same output device). To divide both output-streams, one could attach separate processes to them, adding an output-identifier before sending the output to the client. To solve the buffered I/O problem, the implemented approach makes use of adding code to the source code of the program which must be executed. This code disables all buffering of output, making sure that any output generated by the program is directly send to STDOUT and therefore to the client. Although useable in this situation since the assignment is to write source code, it is not useable in situations where already compiled programs must be executed. For such an environment, one could make use of the non-standard library 'Comm.pl' which emulates a standard tty. A future implementation should therefore make use of both adding identification information to STDOUT and STDERR, while solving buffered I/O problems at the server level.
Security at this level is extremely difficult. First, programs are executed under one username, making it difficult to allow a student to run programs only in their own resource space. Next, malicious code could be included in the programs, resulting in unexpected and possibly unwanted behaviour. The last problem is not new to such an environment, and does also exist in a normal situation. The first problem is generally solved by supplying each student with his or her own username and restrictions to the system. This requires a far more sophisticated approach than described in this thesis. For instance, the Telnet protocol could be implemented, resulting in running programs under the student's username.

7.15 Future enhancements

This part focuses on two related aspects: implementing security and enhancement of the 'tunnel'.
Clearly, the current system provides only a low level of security. Security enhancement is needed in at least three different areas: the connection between the client and the tunnel, the connection between the tunnel and the server and the virtual connection between the client and the server. The first two have to do with providing a secure connection to enable user identification, the last one has to do with preventing a user from harming both the remote system as a whole as well as harming resources used by other students. A first step towards a more secure environment would be to check a user's identity at the tunnel, instead of the server. This is a logical step, since this tunnel is located at the machine the web server is also located on. This machine therefore already contains the username/password combinations necessary for securing the World Wide Web environment on that machine. If this is a Unix environment, the tunnel could for instance make use of the same password file in use by the Web server. The next logical step would therefore seem to be to place all files on the tunnel machine, using the parallel machine only for compilation and execution of programs. However, this requires detailed knowledge of the files needed at the parallel machine to run a program, which is highly dependent on the source code written by a student. Therefore, such an approach is probably not feasible. This requires the creation of specific user environments on the front-end machine, to which a student has to log on. This then requires the creation of a much more sophisticated server, simulating for instance a Telnet connection. Such an approach would indeed resolve the last security area described above. Future enhancements should

therefore focus on three points: providing a secure connection between client and tunnel, providing a trusted connection between tunnel and server, and providing a Telnet like service on the server. The connection between the tunnel and the server could for instance be based on the Secure Socket Layer protocol developed by Netscape, since this provides basic functionality for a trusted connection based on public and private key encryption. At this moment, the distribution of updated standard Java packages also include classes providing basic public/private key encryption functionality. More research is therefore needed into using these classes in combination with the tunnel.

7.16 Conclusions

The use of the client/server paradigm has been useful to describe the Virtual Computer Lab as it was implemented for this thesis. Currently, the server does little more than providing an interface to the operating system, and the tunnel is merely a means of connecting a client to a server located on a remote machine. Authorization is therefore done at the server. But, since the tunnel resides at the central distance learning machine, authorization should be done by the tunnel thus increasing its importance. Doing this, the tunnel can function as a central point for practical assignments, providing additional service such as giving an applet a description of the preferred environment for the practical assignment. The placement of personal files was and will be a problem of some proportions. Currently, they are placed at the server side since place independence requires a central storage facility. Placing them at the tunnel is however practically impossible, since it is hard to know beforehand what files will be needed by for instance a running process. This would require intelligent understanding of a user's code, and storing information about how all files are related. In the light of this prototype, this was seen as being impractical.
If authorization will take place centrally, the link between tunnel and server should consist of a trusted connection, enabling the server to trust the tunnel's authorization information. A public/private key solution such as the SSL protocol could be used for this.
The use of Java for this environment in the form of applets in a Web browser was not completely satisfactory. The expected problems described earlier were almost all present in a number of test runs, sometimes having too much of an influence on the behaviour of the system. It can be concluded that the 'integration' demand put on the environment was defined too strictly. A less strict definition might allow the use of Java Applications, not limited the way Applets are. Such applications can then make use of a Web browser available at the client side, instead of the other way around as is done in this prototype.

8. Conclusions

8.1 Introduction

This final chapter presents a review of all previous chapters in the light of the questions which should be answered by this thesis. It attempts to bring the various chapters together, to answer both the main question as well as discuss whether or not the goal of this thesis has been reached.

8.1.1 How can information- and communication technology help in changing the educational process?

The answer to this question is presented in the first chapter. Based on a number of problems with the current educational model, distance learning as a combination of ICT and a modern pedagogical approach was described as a possible solution. It was concluded that this project should focus on a point between enrichment and innovation. I believe this choice to be valid, since throughout literature extreme problems with making a leap from enrichment to transformation are stressed. The extreme consequences of such an approach cannot be described beforehand, and present a high risk-factor. The conversation I had with J. Klaasse of the faculty of physics of the University of Amsterdam, and presented in (Mirande, 1994) as one of the people using a convincing example of educational software, proved that even moving from enrichment to innovation based on such an example could result in failure. The solution of creating such a convincing example is actually not much different from the well known prototype approach found in standard software engineering literature. Therefore such an approach was used for this research. The basic steps introduced there have proven to be well useable for answering the remaining questions.

8.1.2 What kind of functionality should the required software offer?

In chapter two, a choice for mild constructivism was made. Finding an answer to the more general question of what pedagogical approach to choose proved to be one of the bottlenecks in the creation of an educational system. The last eight years, discussion about whether to choose constructivism or instructivism has been of major influence on literature. One only has to compare the more instructivist ideas of Jonassen presented in (Jonassen, 1988) with his constructivist ideas presented in (Duffy, 1992) four years later, to see how much these two extremes can differ. The heat of this discussion is well visible in (Duffy, 1992), where constructivists and instructivists discuss the implications of both approaches. However, the more literature I read about these subjects, the more it became clear that currently what is called ‘mild constructivism’ is preferred at least by instructional designers, traditionally the ones most involved in combining technology and education. Based on this perception, and the lack of practical implementations of constructivist approaches, the mild constructivist approach as interpreted by instructional designers was chosen. Whether this choice is questionable from an educational perspective is beyond my knowledge, but literature suggests otherwise. However, a final implementation of a distance learning system might require more knowledge about the implications of constructivism on building ICT environments for the higher knowledge levels.

Since the approach instructional designers currently take towards educational systems is still largely based on the use of theory about cognitive models, most of the theory found in chapter two was based on psychological findings. Though hard to find, the combination of a number of articles and books provided the necessary insight into this matter at a level aimed at

understanding the consequences for software design. The chapter resulted in the description of requirements for the different components needed for the distance learning environment. The most important component found in such software is a conceptual structure of the domain, which forms the basis for most of the remaining components. The main conclusion leading to chapter three is the notion that the retrieval component of an information system is important to support the dynamic structure of a learning system. Furthermore, the importance of a social interaction environment was stressed, as well as the need to provide interactive content with the possibility to make annotations. Broad in its approach, the chapter did result in useable information about the backgrounds of an also broad pedagogic approach towards software development. Although the results were not specifically surprising since they can be found in basic approaches towards educating students, they did provide the necessary backgrounds to understand the impact of current and future choices with respect to enabling learning.

8.1.3 Which components should be present when the system is build using client/server based information retrieval?

Here, the findings of chapter two are taken to a more abstract level. Based on a standard approach towards describing information retrieval systems, a model for distance learning supporting the findings of chapter two was described. The importance of deciding on a student's level of domain knowledge created the need to look into literature combining artificial intelligence and educational systems. This is the domain of Intelligent Tutoring Systems, largely based on Expert System theory, which presented a more familiar area for me. ITS systems are based on dividing the important components found in teaching, enabling easy replacement of any of the components without effecting others. Traditionally aimed at replacing the instructor, the role this person plays in such an environment seems to be neglected. Next to this, the division of components was not complete when compared to the information retrieval approach. Finally, after adding a component for social interaction, a final model was devised, of which the more detailed description was left to following chapters. Especially the level of abstraction of the social interaction component was not satisfactory here, leading to a more detailed description in chapter four. For the rest I can only say that the division in components has both structured the remaining chapters and provided a tool for reviewing existing systems. Especially the last function was important, since most systems only implemented part of the requirements, and on the surface seemed to differ a lot.

8.1.4 How should the server components be build?

This question is basically a logical consequence of the model described before. ITS literature describes the importance of using student assessment to adapt the environment to individual students, but lacked in a detailed description of such assessment. At first, this resulted in the reading of standard artificial intelligence literature, combining this with theory about analyzing test items found in (Dousma, 1995). However, a question posed to professor C. Jaffe of the Center for Advanced Instructional Media directed me towards the field of computer-based adaptive testing. Furthermore, it turned out that the ARIES research group in Canada had done a lot of research on combining conceptual domain structures, Bayesian Belief Networks and testing theory. Especially the thesis written by Jason Collins (Collins, 1996) proved to be useful in directing me to literature about implementing such a combination. However, the results of this chapter were a bit disappointing with respect to testing higher levels of knowledge, for which no readily available tools seem to exist. Future research should therefore focus on combining qualitative and quantitative approaches towards this problem.
Another important part found at the server side is an Answer/Question Knowledge-base. A basic system can be found in (Ackerman, 1993) but especially the Knowledge Tree extension

proved to be useful for this project. Such a system is also based on using the Domain Knowledge module for structuring discussions. Finally, a simple annotation system could make use of the same approach as the Knowledge Tree.

8.1.5 Can the World Wide Web theoretically support such an environment?

Theoretically, all needed components can be provided by using the World Wide Web. Concern was expressed for the usage of Java Applets within a distance learning environment, which later turned out to be valid. For the rest, it was expected that the World Wide Web would provide a platform too much focused on presentation, instead of dividing applications across a network, creating the need to use Java Applets.

8.1.6 What kind of functionality do existing World Wide Web based systems offer?

With respect to multimedia it was expected that current advances in providing higher bandwidth to users would result in the possibility to use higher quality multimedia in distance learning environments. Without increasing bandwidth at the server side, a combination of streaming media and multicasting technology was thought to ultimately lead to distributing the complete distance learning system across several servers. Falling outside of the direct scope of this thesis, appendix A describes some ideas about such an approach. Finally, chapter six describes the current state-of-the-art with respect to distance learning environments on the World Wide Web. Within a half year, the number of professionally organized distance learning environments on the World Wide Web has grown considerably. However, such systems generally lack in an approach towards adapting the environment to a student's knowledge level, as well as in an integrated social interaction environment. The general use of an adaptive toolbar leaded to the need for a more detailed description in the following chapter. In general, the hypertext environment of the World Wide Web seemed to favor a book-like approach towards structuring the contents, of course clashing with conclusions from chapter two, indicating that this is an unpreferrable implementation.
Chapter seven provides a more detailed description of two components: the toolbar and the social interaction environment. It turned out that the first was easy to implement, while the latter was too complex to be included in the implemented prototype described in this thesis. The approach used in Knowledge Tree towards structuring such an environment was seen as being easy to integrate with the system described thus far, due to its focus on the use of a conceptual domain structure. Since this thesis does not provide more information on the subject than a discussion about the functionality of such an environment, more research will be needed for this before building the final system. So far, no limitations presented by using the World Wide Web for social interaction were identified. In fact, this chapter shortly described the use of an SQL database to store a conceptual structure. The importance of such a structure for the system as a whole requires a detailed description of using such a relational database for storage, which is not presented in this thesis.

8.1.7 How can a Virtual Computer Lab be implemented?

This has been a very broad question, which is due to the fact that beforehand not much was known about the implementation of such a lab. Answering this question has therefore lead to the actual implementation of a virtual computer lab prototype, leaving a lot of unresolved questions. The question should not be seen as a quest for knowledge about some development procedure, but rather as a question about technical aspects of such an implementation. Chapter

five already provided some concerns with respect to the usage of Java Applets, and most of those concerns became reality when implementing this prototype. However, the approach chosen presented a possible way to move the assignment towards time and place independence. Based on initial experience, it was concluded that a future implementation should provide the tunnel with more functionality, while researching ways to make the system secure. The use of the Secure Socket Layer model was presented as one way to enhance security. The choice for Java Applets has been questioned, favoring the also platform independent Java Applications. However, choosing such an approach requires the student to install the complete Java environment while moving somewhat away from the demand for integration of the different components.

8.2 Concluding remarks

Broadly spoken, I think this thesis has provided an answer to the main question of which components should be present in an integrated Distance Learning system for the specific summer course, and how these components can be implemented using the World Wide Web as a platform. By doing this, it has provided HPCnet with insight into how to use the World Wide Web to provide students with a time and place independent way of participating in courses on High Performance Computing. Future work in this area probably requires the help of experts at all of the involved areas, including education, artificial intelligence, hypermedia and networking. The move towards innovation also requires a structured approach, with active involvement of all parties concerned. I hope this thesis will prove to be useful for the next steps involved in creating the integrated distance learning environment.

9. Used Literature

(Ackerman, 1993) Ackerman, M.S., 'Answer Garden: a tool for growing organizational memory', Phd thesis Massachusetts Institute of Technology, 1993

(Ambrosio, 1991) D'Ambrosio, B., Li, Z. 'Efficient Inference in Bayes Nets as a Combinatorial Optimization Problem' International Journal of Approximate Reasoning, Vol 11 No 1, 1991, pages 55-81

(Berghout, 1995) Berghout, E.W., van Irsel, H.G.P., Meertens, F., 'Legitimeren van IT-inversteringen', Eksbit Werkgroep Information Economics, 1995

(Berners-Lee, 1996) Berners-Lee, T., Fielding, R., Frystyk, H., 'Hypertext Transfer Protocol – HTTP/1.0', Network Working Group, Request For Comments: 1945, 1996

(Brailsford, 1997) Brailsford, T.J., Davies, P.M.C., Scarborough, S.C., Trewhella, W.J., 'Knowledge Tree: putting discourse into computer-based learning', in 'ALT-J', Volume 5, Number 1, Association for Learning Technology, 1997, pages 19-26

(Cellear, 1997) Cellear, D., King, T., 'Using computer-based tests for information science', in 'ALT-J', Volume 5, Number 1, Association for Learning Technology, 1997, pages 27-32

(Collins, 1996) Collins, J.A., 'Adaptive Testing with Granularity', thesis University of Saskatchewan, 1996

(Collis, 1996) Collis, B., 'Tele-learning in a Digital World - The Future of Distance Learning', International Thomson Computer Press, London, 1996

(Conklin, 1987) Conklin, J., 'Hypertext: An Introduction and Survey', IEEE Computer, 1987, pages 17-41

(Dick, 1992) Dick, W. 'An Instructional Designer's View of Constructivism', in: 'Constructivism and the Technology of Instruction' edited by T.M. Duffy and D.H. Jonassen, Lawrence Erlbaum Associates inc., New Jersey, 1992, pages 91-98

(Dousma, 1995) Dousma, T., Horsten, A., 'Tentamineren', Wolters-Noordhoff, Groningen, 1995

(Duffy, 1992) Duffy, T.M., Jonassen, D.H., 'Constructivism and the Technology of Instuction', Lawrence Erlbaum Associates, New Jersey, 1992

(Duguet, 1995) Duguet, P., 'Education: Face-to-face or Distance?', in: The OECD Observer, no. 194, 1995, pages 17-20

(Flanagan, 1996) Flanagan, D., 'Java in a nutshell', O'Reilly & Associates Inc., Sebastopol, 1996

(Fuji, 1996) Fuji, T., Tanigawa, T., Kozeni, M., Inui, M., Saegusa, T., 'A Case-Based Approach to Collaborative Learning for Systems Analyst Education', in: 'Intelligent Tutoring Systems, 3rd International Conference' edited by C. Frasson, G. Gauthier and A. Lesgold, Springer, Berlin, 1996, pages 177-186

(Gastkemper, 1997) Gastkemper, F., Nijhuis, G.G., Collis, B., Vleugel, B., 'Met teleleren naar beter onderwijs op maat', in: 'De virtuele organisatie', edited by: R. van Dael and C. Metselaar, Kluwer Bedrijfsinformatie, Deventer, 1997, pages 77-92

(Geoghegan, 1994) Geoghegan, W.H., 'What ever happened to instructional technology?', 22nd Annual Conference of the International Business Schools Computing Association, IBM, Maryland, 1994

(Gundavaram, 1996) Gundavaram, S., 'CGI Programming on the World Wide Web', O'Reilly & Associates Inc., Sebastopol, 1996

(Halasz, 1994) Halasz, F., Schwartz, M., 'The Dexter Hypertext', Communications of the ACM, Vol.37, No.2, 1994, pages 30-39

(Hämäläinen, 1996) Hämäläinen, M., Whinston, A.B., Vishik, S., 'Electronic Markets for Learning: Education Brokerages on the Internet', Communications of the ACM, vol.39, No.6, 1996, pages 51-58

(Hambleton, 1991) Hambleton, R.K., Swaminathan, H., Rogers, H.J., 'Fundamentals of Item Response Theory', Sage Publications, London, 1991

(Harold, 1997) Harold, E.R., 'Java Network Programming', O'Reilly & Associates Inc., Sebastopol, 1997

(Huang, 1996) Huang, S.X., 'A Content-Balanced Adaptive Testing Algorithm for Computer-Based Training Systems', in: 'Intelligent Tutoring Systems, 3rd International Conference' edited by C. Frasson, G. Gauthier and A. Lesgold, Springer, Berlin, 1996, pages 306-314

(Hunt, 1992) Hunt, C., 'TCP/IP Network Administration', O'Reilly & Associates inc, Sebastopol, 1992

(Ives, 1996) Ives, B., Jarvenpaa, S.L., 'Will the Internet Revolutionize Business Education and Research?', Sloan Management Review, Spring 1996, pages 33-40

(Jansen, 1997) Jansen, E.A., 'Multimedia in scholingssituaties - een beschrijving van gebruik en implementatie', thesis Rijksuniversiteit Groningen , 1997

(Jonassen, 1988) Jonassen, D.H., 'Instructional designs for microcomputer courseware', Lawrence Erlbaum Associates, New Jersey, 1988

(Jonassen, 1992) Jonassen, D.H., 'Evaluating Constructivistic Learning', in 'Constructivism and the Technology of Instruction' edited by T.M. Duffy and D.H. Jonassen, Lawrence Erlbaum Associates inc., New Jersey, 1992, pages 137-148

(Joyce, 1984) Joyce, B., Weil, M., 'Strategieen voor onderwijzen' (Dutch edition of 'Models of teaching'), Van Walraven bv, Apeldoorn, 1984

(Kalakota, 1996) Kalakota, R., Whinston, A.B., 'Frontiers of electronic commerce', Addison-Wesley publishing company inc, 1996

(Kanselaar, 1991) Kanselaar, G., De Tombe, D.J., 'Courseware evaluatie', Academisch Boeken Centrum, de Lier, 1991

(Kearsly, 1987) Kearsly, G., 'Artificial Intelligence and Instruction, Applications and Methods', Addison-Wesley Publishing Company, Massachusetts, 1987

(Khuwaja, 1996) Khuwaja, R., Desmarais, M., Cheng, R., 'Intelligent Guide: Combining User Knowledge Assessment with Pedagogical Guidance', in: 'Intelligent Tutoring Systems, 3rd International Conference' edited by C. Frasson, G. Gauthier and A. Lesgold, Springer, Berlin, 1996, pages 225-233

(Kirschner, 1995) Kirschner, P., Hermans, H., De Wolf, H., 'Onderwijsvernieuwing en informatietechnologie', Educatieve Partners Nederland bv, Houten, 1995

(Kumar, 1996) Kumar, V., 'Mbone - Interactive Multimedia on the Internet', New Riders Publishers, Indiana, 1996

(Leavitt, 1996) Leavitt, M.O., 'Virtual U', in: MultiVersity, winter 1996, pages 12-15

(Lotus, 1996) Lotus, 'Distributed Learning: Approaches, Technologies and Solutions - White Paper', August 1996

(Luger, 1998) Luger, G.F., Stubblefield, W.A., 'Artificial Intelligence - Structures and Strategies for Complex Problem Solving', Addison Wesley Longman inc, Reading, 1998

(Mark, 1992) Mark, M.A., Greer, J.E., 'Evaluation Methodologies for Intelligent Tutoring Systems', ARIES Laboratory, Saskatoon, 1992

(Mast, 1995) Mast, C. Van der, 'Developing Educational Software, integrating disciplines and media', Proefschrift Technische Universiteit Delft, 1995

(Merril, 1992) Merril, D., 'Constructivism and Instructional Design', in: 'Constructivism and the Technology of Instruction' edited by T.M. Duffy and D.H. Jonassen, Lawrence Erlbaum Associates inc., New Jersey, 1992, pages 99-114

(Minoli, 1996) Minoli, D., 'Distance Learning Technology and Applications', Artech House inc., Norwood, 1996

(Mirande, 1994) Mirande, M.J.A., 'De rol van de computer in het hoger onderwijs', in: 'De kwaliteiten van computer ondersteund onderwijs' edited by M.J.A. Mirande, Dick Coutinho, Bussum, 1994, pages 15-39

(Nicol, 1995) Nicol, D., 'Footsteps: Trail-blazing the Web', The Third International World-Wide Web Conference, 1995

(Park, 1987) Park, O., Perez, R.S., Seidel, R.J., 'Intelligent CAI: Old Wine in New Bottles, or a New Vintage?', in: 'Artificial Intelligence and Instruction, Applications and Methods' edited by G. Kearsly, Addison-Wesley Publishing Company, Massachusetts, 1987, pages 11-46

(Perkins, 1991) Perkins, D.N., 'Technology Meets Constructivism: Do They Make a Marriage?', in: 'Constructivism and the Technology of Instruction' edited by T.M. Duffy and D.H. Jonassen, Lawrence Erlbaum Associates inc., New Jersey, 1992, pages 45-55

(Pilot, 1994) Pilot, A., 'Potenties van de computer in het hoger onderwijs', in: 'De kwaliteiten van computer ondersteund onderwijs' edited by M.J.A. Mirande, Dick Coutinho, Bussum, 1994, pages 40-54

(Pressman, 1992) Pressman, R.S., ‘Software engineering, a practitioner’s approach’, McGraw-Hill, New York, 1992

(Rice, 1993) Rice, R., ‘Multi- en interactieve media, toepassingen binnen organisaties’, in: multimedia tussen hope en hype, Otto Cramwinckel uitgever, Amsterdam, 1993

(Richards, 1996) Richards, S., ‘Educational Software design: applying models of learning’, in ‘ALT-J’, Volume 4, Number 3, Association for Learning Technology, 1996, pages 17-20

(Savetz, 1996) Savetz, K., Randall, N., Lepage, Y., ‘Mbone - Multicasting tomorrow’s Internet’, IDG Books WorldWide inc, Foster City, 1996

(Schank, 1994) Schank, R.C., ‘Active Learning through Multimedia’, IEEE MultiMedia, Spring 1994

(Somekh, 1996) Somekh, B., ‘Designing software to maximize learning’, in ‘ALT-J’, Volume 4, Number 3, Association for Learning Technology, 1996, pages 4-16

(Tanenbaum, 1996) Tanenbaum, A.S., ‘Computer Networks’, Prentice-Hall International inc, 1996

(Trevino, 1992) Trevino, L.K., Daft, R.L., Lengel, R.L., ‘Understanding managers’ media choices: a symbolic interactionist perspective’, Reader CS Multimediatoepassingen 1994/1995

(Turban, 1993) Turban, E., ‘Decision Support and Expert Systems’, Macmillan Publishing Company, New York, 1993

(Venners, 1997) Venners, B., ‘Inside the Java Virtual Machine’, McGraw-Hill Companies Inc., 1997

(Wallach, 1987) Wallach, B., ‘Development Strategies for ICAI on Small Computers’, in: ‘Artificial Intelligence and Instruction, Applications and Methods’ edited by G. Kearsly, Addison-Wesley Publishing Company, Massachusetts, 1987, pages 305-322

(Wasson, 1996) Wasson, B., ‘Instructional Planning and Contemporary Theories of Learning: Is this a Self-Contradiction?’, in: ‘Proceedings of the European Conference on Artificial Intelligence in Education’ edited by P. Brna, A. Paive and J. Self, Colibri, Lisbon, 1996, pages 23-30

(Yacef, 1996) Yacef, K., Alem, L., ‘Student and Expert Modelling for Simulation-Based Training: A Cost Effective Framework’, in: ‘Intelligent Tutoring Systems, 3rd International Conference’ edited by C. Frasson, G. Gauthier and A. Lesgold, Springer, Berlin, 1996, pages 614-622

(Yeh, 1996) Yeh, P., Chen, B., Lai, M., Yuan, S., ‘Synchronous navigation control for distance learning on the Web’, in: Computer Networks and ISDN Systems 28 (1996) 1207-1218

10. Appendix A: Towards a distributed Distance Learning system

One of the remaining questions of chapter 6 is whether to use the push-model of Splitters, or the pull-model of Proxies for a Distributed Distance Learning system. At the moment, content of Web pages cannot be distributed using Splitters, and no Proxy server exists to cache RealMedia files. However, both situations are possible and not too difficult to implement. The reason why RealMedia is distributed using Splitters has a lot to do with the fact that it aims at providing live broadcasts as efficiently as possible. Proxies on the other hand cannot have a constant feed from a main-server, since no such thing exists on the Internet as a whole. A proxy does not know which documents are needed, and no Web-server on the Internet knows which proxy server will request a document next.
Using a proxy to distribute RealMedia files would result in something like the following situation:

- A RealPlayer makes a TCP connection with the proxy;
- The proxy makes a TCP connection with the RealServer;
- Based on bandwidth negotiation, the RealServer decides which RealMedia file to use;
- The Proxy checks the name and date of that file with the files in its cache;
- If the file is available at the Proxy, it will function as a RealServer, if not, it will ask the RealServer to send the file with the highest speed possible using TCP and starts transmitting it to the RealPlayer while caching the file.

Although this seems plausible, problems will occur when a user skips a part of the RealMedia file and jumps to a position which has not yet been downloaded to the Proxy. The Proxy will then have to ask the RealServer to start sending the requested part of the RealMedia file, needing a mechanism to put pieces of the RealMedia file together while preventing to open multiple streams to the RealServer. A next request from a client will result in a situation where parts of the RealMedia file are transmitted directly from the cache, and parts are requested from the RealServer. As long as the international link through which Proxy and RealServer are communicating can provide enough bandwidth to send streaming video to the client, this schema could work. However, if this is not the case, the problem has just been extended to the Proxy. Therefore, a time-critical application such as a RealMedia system is based on an ‘available-before-needed’ basis. A less time-critical situation where Web pages are requested doesn’t suffer too much from delay times, and can therefore use a ‘request-when-needed’ model.
The push model should however work alright in a situation where Web pages are requested. One of the main problems would be located at the main server which has to know when a file is changed.
Either way, using a distributed approach reduces bandwidth problems, and distributes the load on servers. Currently, in an environment which makes heavy use of both standard Web pages as well as RealAudio and RealVideo, the system would look like Figure 1.

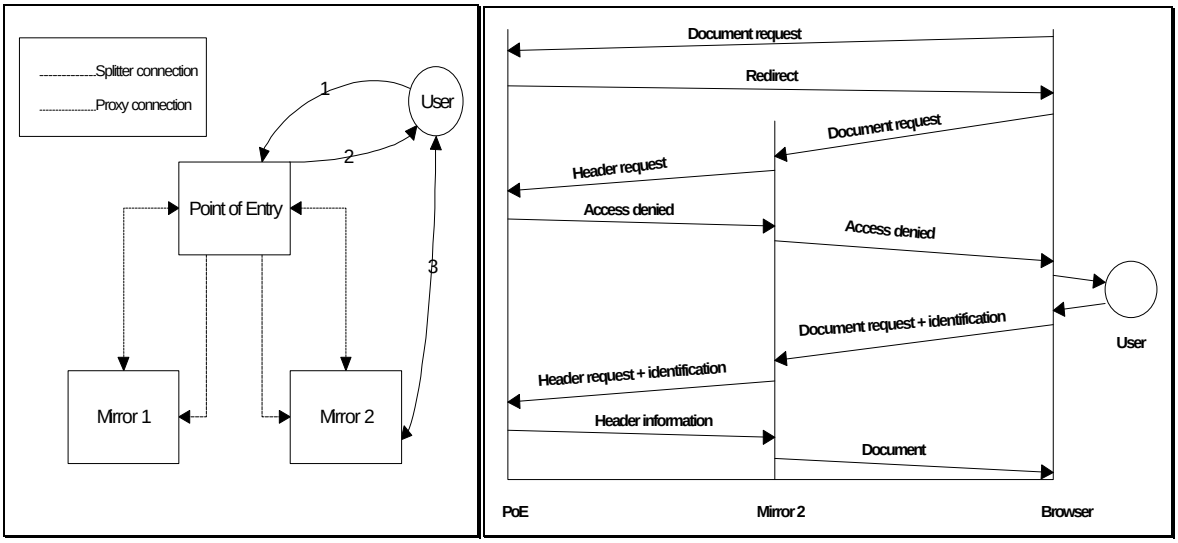


Figure 1 - Distributed Distance Learning System with Client/Server communication

The 'Point of Entry' is the main server containing all original documents and RealMedia files. Mirrors 1 and 2 consist of a Web-Server with proxy-capabilities and a RealMedia Splitter. The user has a Web-browser containing the (free) plug-in required to view RealMedia streams. The user requests a document from the 'Point of Entry' (PoE). This machine decides, based on the ip-address of the client, to which mirror server the user should be connected (Mirror 2). It then sends a redirect message back to the user's browser, which requests the document from Mirror 2. This server tries to get the header information of the document from the PoE server, but gets an 'Access Denied' which it returns to the browser. The browser then asks the user to provide the right username and password, which it attaches to the next request of the document. Mirror 2 adds the identification information to its request to the PoE server, which checks the given username and password. If they are valid, it returns the header information. Mirror 2 is now able to compare the document in its cache with the one on the PoE server. If they match, the local document is returned to the browser. The browser has stored the username and password, and automatically attaches them to any subsequent request to Mirror 2, if they seem needed. Requesting RealMedia files follows the same principle.
Such a system could be build using available servers at this point in time.