HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Laboratory of Acoustics and Audio Signal Processing

Martti Rahkila

Agent-based Method for Self-study Interactive Web-based Education

Thesis submitted in partial fulfillment of the requirements for the degree of Licentiate of Science in Technology.

Espoo, May 17th, 2006

Supervisor: Professor Matti Karjalainen

HELSINKI UNIVERSITY OF TECHNOLOGY		ABSTRACT OF THE LICENTIATE'S THESIS

Author:	Martti Rahkila		
Name of the thesis:	An Agent-based Method for Self-study Interactive Web-based Education		
Date:	May 17th, 2006	**Number of pages:**	67
Department:	Electrical and Communications Engineering		
Professorship:	S-89		
Supervisor:	Prof. Matti Karjalainen		

This thesis deals with computer-based education of acoustics and digital signal processing. The focus throughout the thesis is on interactive, self-study web-based applications even though many issues are of more general nature as well. The emphasis is especially on describing interactivity while using educational applications and the use of log information for evaluation of learning.

The goal for the thesis has been to develop a web-based solution for audio signal processing education with emphasis on advanced, intelligent interactivity.

The basis for this interactivity is the double agent architecture for web applications. The architecture allows the control of the interaction process by means of logs and using them as a basis for content adaptation. Furthermore, the novelty of this method is its applicability to evaluation of learning. The log information, provided by the architecture, can be used for on-line evaluation of users' requests and thus provides means for content adaptation. Furthermore, the log information can also be post-processed and used for off-line evaluation of the learning process by both teachers as well as students themselves. The latter has also pedagogical importance supporting the development of self-reflection and metacognitive skills.

Keywords: World Wide Web (WWW), Computer Based Education (CBE), interactive web applications, software agents, audio technology, digital signal processing, evaluation of learning, content adaptation

TEKNILLINEN KORKEAKOULU LISENSIAATINTYÖN TIIVISTELMÄ

Tekijä:	Martti Rahkila		
Työn nimi:	Ohjelmistoratkaisu vuorovaikutteisiin WWW-pohjaisiin itseopiskelusovelluksiin		
Päivämäärä:	17.05.2006	**Sivuja:**	67
Osasto:	Sähkö- ja tietoliikennetekniikka		
Professuuri:	S-89		
Työn valvoja:	Prof. Matti Karjalainen		

Työ käsittelee akustiikan ja digitaalisen signaalinkäsittelyn tietokonevusteista opetusta. Painopiste on kautta linjan vuorovaikutteisissa, itseopiskeluun tarkoitetuissa WWW-sovelluksissa, vaikka työssä käsitellään aihepiiriä laajemminkin. Työ keskittyy erityisesti vuorovaikutteisuuden kuvaamiseen ohjelmistoa käytettäessä sekä käytön yhteydessä saatavan lokitiedon hyödyntämiseen oppimisen arvioinnissa.

Työn tavoitteena on ollut kehittää etenkin audiosignaalinkäsittelyyn soveltuva ohjelmistoratkaisu hyödyntämällä nykyaikaista WWW-teknologiaa. Avainasemassa on ollut pyrkiä kehittämään nimenomaan älykkään vuorovaikutteisuuden kuvaamiseen sopiva ratkaisu.

Vuorovaikutteisuuden perustana toimii WWW-sovelluksille suunniteltu ns. kaksoisagenttiarkkitehtuuri. Arkkitehtuuri mahdollistaa vuorovaikutusprosessin ja sisällön ohjaamisen lokitiedon avulla. Menetelmän keskeisin uusi oivallus on tämän lokitiedon käyttö oppimisen arviointiin. Lokitiedot mahdollistavat sekä reaaliaikaisen käyttäjien toimien analysoinnin ja sisältömateriaalin muokkaamisen tämän mukaisesti, että jälkikäteen tapahtuvan oppimisprosessin arvioinnin. Arviointia voivat jälkikäteen tehdä sekä oppimateriaalin tuottajat (opettajat) että oppimateriaalin käyttäjät (opiskelijat) itsearviointina. Tällä on oppimisen kannalta tärkeä merkitys ns. metakognitiivisten taitojen kehittymiselle.

Avainsanat: World Wide Web (WWW), tietokoneavusteinen opetus, vuorovaikutteiset www-sovellukset, ohjelmistoagentit, ääniteknologia, digitaalinen signaalinkäsittely, oppimisen arviointi

Preface

So, it took a little longer than I expected... This project dates back all the way to the mid 1990's. At the time, the web was something new but already showed great potential as a tool that might bring the Internet to homes and offices. Which it certainly did. But back then, it was wild frontiers and nobody knew what kind of technologies would evolve and survive within the next decade or what purposes would this marvelous tool be used for.

The development has been rapid ever since. However, amazingly, I'm still facing some of the same technical problems as ten years ago. But to me, it just proves the value of this research: the goals have been high and the technical problems challenging enough for these questions to survive the technological revolution that has changed the world around it.

To me personally, the path has been fruitful: like all good research, it has answered questions but also opened up new questions. I have been able to watch and study the development of the web both in technical terms as well as contents wise. But most importantly, my knowledge and understanding of learning itself has grown and become deeper throughout the project. Which was the reason I got interested in this work in the first place.

I wish to thank the Helsinki University of Technology as well as Tekniikan Edistämissäätiö for supporting this work during the first four years. Furthermore, I'd like to thank Petter ja Margit Forsströmin säätiö for their scholarship.

Furthermore, I'd like to express my warmest thanks to my supervisor, professor Matti Karjalainen. I have had a pleasure working with him for several years, not only on this project but in many others as well. I truly admire his enthusiasm and open mind.

I'd also like to thank professor Lauri Savioja, who has been of help especially during the last few months of finalizing this thesis. I have known Lauri for several years as well and he is one of kindest persons I have ever met. His comments have been most valuable and encouraged me to try to improve the thesis.

There are numerous wonderful people that I have met during the years at the TKK Acoustics lab. It would be meaningless to try to list all your names: the list would be longer than this thesis and, besides, you know who you are anyway. You have all taught me a lot for which I thank you for. The same goes for my loved ones as well.

But it is not over yet! There are still many problems to be solved, many questions to be answered. Hopefully, the project continues.

Helsinki, May 17th, 2006

Martti Rahkila

Contents

List of Abbreviations

AH	Adaptive Hypermedia
AJAX	Asynchronous JavaScript and XML
API	Application Program Interface
ASP	Active Server Pages
CAI	Computer Assisted Instruction
CBE	Computer Based Education
CBI	Computer Based Instruction
CBL	Computer Based Learning
CBT	Computer Based Training
CGI	Common Gateway Interface
CMI	Computer Managed Information
CMS	Content Management System
CSCL	Computer Supported Collaborative Learning
CSLR	Computer Supported Learning Resources
CSS	Cascading Style Sheets
DAML	Darpa Agent Markup Language
DHTML	Dynamic HTML
DOM	Document Object Model
DSP	Digital Signal Processing
ECMA	European Computer Manufacturers Association
GUI	Graphical User Interface

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPR	Reliable HTTP
J2EE	Java2 Enterprise Edition
JRE	Java Runtime Environment
JSP	Java Server Pages
JVM	Java Virtual Machine
MP3	MPEG-1 Layer III
MPEG	Moving Pictures Expert Group
PDF	Portable Document Format
RDF	Resource Description Framework
RFC	Request For Comments
SMIL	Synchronized Multimedia Integration Language
SQL	Structured Query Language
SVG	Scalable Vector Graphics
UML	Unified Modeling Language
URI	Unified Resource Identificator
URL	Unified Resource Locator
W3C	World Wide Web Consortium
WBE	Web Based Education
WBT	Web Based Training
WSDL	Web Service Definition Language
WS-RM	Web Services Reliable Messaging
WWW, web	World Wide Web
XML	Extensible Markup Language
XMLP	XML Protocol

Chapter 1

Introduction

This thesis discusses web-based education in acoustics and digital signal processing. The focus throughout the thesis is on the technical design of interactive self-study applications even though many issues discussed are also of more general nature. The key issue and novelty of the thesis is the evaluation of learning based on logging of users' actions. In order to do this, an intelligent, agent-based system architecture is presented.

The thesis is largely based on conference articles written by the author [95] [94] [93] [97] [96].

1.1 Project background

The background of this project dates all the way back to years 1996-1997. At the time, the web was something completely new: a hypermedia platform showing great promise but yet immature both technically and contents wise. The number one browser was Netscape 2.0 which presented many technological innovations [84], later to become more or less open web standards. Examples of such technologies are JavaScript [49] and support for Java Applets [106].

Technologically this was the beginning era of the Internet revolution. Nobody could tell which of these new technologies would be useful and what they would be used for eventually. Every new browser version included new features and technologies and the development of server-side software was equally fast. It took about 5 years until the situation stabilized technologically and W3C (World Wide Web Consortium) and other open standards became the real backbone of the web. During that time, also the new media business was born, and almost destroyed as well by the wild market expectations. However, during that time the Internet grew out from universities and research institutes conquering the business world as well as homes and ordinary consumers.

Back in 1996, I had just finalized my Master's Thesis [92] introducing a stand-alone CBE (Computer Based Education) application for the fundamentals of digital signal processing (DSP). The application was built with the Common Lisp based QuickSig [58] DSP environment developed at the Helsinki University of Technology Laboratory of Acoustics and Audio Signal Processing. The practical setup included an Apple Macintosh workstation located at the laboratory and students reserved time for using the application.

Therefore, the obvious starting point for this project was to investigate, if a similar application could be implemented and distributed using the web instead of the stand-alone workstation. Also higher goals were set, for instance independence of the browsers and platforms as well as building an architecture that would allow development of different kinds of acoustics and DSP related CBE applications like the QuickSig environment had done.

1.2 Goals

The goal for the thesis was to develop a solution for computer-based education of digital signal processing using the World Wide Web (WWW, web) as a platform. The emphasis is on the process of interaction between the user (student) and the application. Understanding and supporting learning has been the key driving factor and goal for the solution.

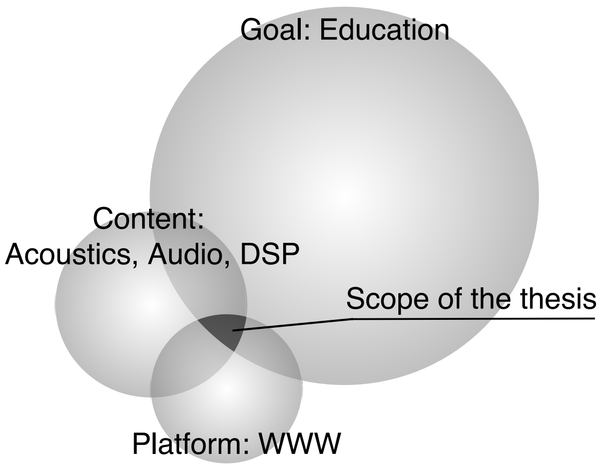


Figure 1.1: The background and scope of the thesis

1.3 Structure of the thesis

Figure 1.1 represents the background and scope of the thesis. Three major areas are considered: pedagogical background to provide a solid ground for learning and studying, acoustics and audio signal processing as the content of the educational material with discussion of their special requirements and, finally, web technology as the platform for implementation. Based on all three, a method for creating and providing such content is proposed.

Therefore, the thesis first presents the pedagogical background followed by discussion of acoustics and audio signal processing as educational content and technical requirements. After these, the field of web technology is introduced and discussed from the point of view of interactive applications. Lastly, a solution proposal is presented with discussion of implementation issues as well as conclusions and guidelines for future work.

The thesis has been written with university level education in mind. Especially teachers of acoustics, digital signal processing and similar fields hopefully will find this work of interest. However, even though the proposed architecture has been designed primarily for educational applications, similar features such as log-driven content adaptation may be of interest in other application areas as well.

Chapter 2

Pedagogical Background

Education: teaching, studying and learning is in itself a wide area of research and practice. There are numerous ways to teach and study, but learning is the ultimate key and goal for which they all aim for. This thesis deals with only a narrow field, web-based self-study education.

2.1 Theories of Learning

Throughout the years, many theories of learning have been developed [63]. These theories emphasize and explore different psychological aspects: after all, learning is indeed fundamental to the human nature. However, learning is still somewhat a mystery, at least philosophically speaking.

Based on pedagogical studies and several years of teaching experience, the so-called cognitive-constructive view of learning [23], [71] has been the guideline for this work. This theory claims that learning is constructing: new things are constructed using previous knowledge as "building blocks". The learner tries to analyze, describe and evaluate a new concept by searching for already learned things and using them to construct the new one. This learning process inside one’s head is the cognitive part of the theory, and the goal for all types of education is to start and maintain this process.

However, in the cognitive-constructive view of learning, also social contacts play an important role. The construction process is fed and accelerated by discussing the topic with others because this way one can reflect his/her thoughts. In fact, the theory of constructionism asserts that constructivism occurs especially well when the learner is engaged in constructing something for others to see [31].

Naturally, learning theories have an important role in the development of Computer Based Education (CBE). In many cases, CBE has been a useful test method for learning theories. Recently, the use of web-based learning environments and collaborative learning tools have been used for exploring especially the social constructivistic theory of learning. In fact, in the field of Computer Supported Collaborative Learning (CSCL) many tools such as Moodle [32] have been designed especially that in mind. CBE has also influenced the development of learning theories. For example, the development of the FLE3 [117] tool has resulted also in a pedagogic model of progressive inquiry learning (PI model) [51]. Discussion of the constructivistic learning theory and CBE can be found in an article by Gance [45].

In this work, the focus is on self-study material and the human-computer interaction. Therefore, the social aspects of constructivistic learning theory have been slightly neglected. On the other hand, more emphasis has been put on the content design questions and evaluation of learning.

2.2 Computer Based Education

Computer Based Education (CBE) can be defined as the use of computers to help people learn. However, this definition is too broad for most practical purposes. The commonly used terms include Computer Assisted Instruction (CAI) which is typically limited to the use of computers in the actual instructional process. Computer Managed Information (CMI) includes for example testing and Computer Supported Learning Resources (CSLR) is a common name for learning resources such as databases as well as collaborative learning environments. Distance learning covers all of these as long as physical presence of the learner is not required, however, nowadays the term is mostly used for on-line learning resources, collaborative tools and making audio and video recordings of lectures and exercises.

Synonyms for CBE include Computer Based Training (CBT), Computer Based Learning (CBL) and Computer Based Instruction (CBI). Nowadays also the terms Web Based Education (WBE), Web Based Training (WBT) etc. are used. The term eLearning is an umbrella term for all Internet-based education.

2.2.1 CBE History

The history of computer-based education (CBE) is almost as long as the history of computers. Ever since the early days in the history of computers, educational applications have been developed [78], [63]. For instance in the 1960’s Patrick Suppes developed various CAI applications for mathematics and other subjects. He designed highly structured systems featuring learner feedback, lesson branching and student record keeping [28].

In the 1970’s, the era of the mainframe computers, the PLATO system developed at the University of Illinois grew to become an online community and offered hundreds of tutorials and drill-and-practice programs [28] through simple text terminals.

In the 1980's, the invasion of personal computers into homes and offices allowed also the growth of multimedia-based CBE. Numerous tools and applications were developed for various purposes [57],[104]. Hypertext and hypermedia [26] became reality. Design methods for educational applications were developed and CD-ROMs became the most popular media for distributing these applications.

However, the growth of the Internet and especially the World Wide Web (WWW, web) in the mid 1990's, launched a new era in CBE. Compared to earlier, the Hypertext Markup Language (HTML) [121] was an easy way to start developing hypermedia applications and the Internet made it possible to distribute and access those applications from anywhere. However, the CD-ROM-based approach allowed more control over the computer resources and stable run-time environments compared to Internet technologies. Even today the web still does not provide similar technical establishment for CBE than CD-ROM-based solutions, but despite these limitations, the web has become a de-facto hypermedia platform due to its easiness of use and sharing of resources over the network.

2.2.2 Classifications of CBE

One way of classifying CBE is through the educational field of interest. Because the ways how computer technology can be used for educational purposes, depend on the content, it is natural that these methods have been developed and studied within this field of interest. Therefore, we may talk about computer-based mathematics education, computer-based language education etc. In this case, the term does not classify the CBE methods themselves but restricts the methods to ones appropriate in that field. For instance the CBE methods and their requirements for mathematics education are quite different from those of language education. Another example: this thesis deals with acoustics and DSP education, both fields presenting their own requirements for meaningful use of CBE. This classification by the content field is especially important, because in CBE, the content defines which methods, content types and, most importantly, what kind of interaction is appropriate and needed.

Another classification is to categorize the CBE by type or method such as drill, game, modeling, simulation, problem solving etc. This categorization is independent of the topic and methods can be used in various fields of interest. However, typically some methods are more suitable for some topics than the others. For instance drill and practice have a long tradition in the language education whereas simulations, modeling and problem solving are standard methods engineering education, computer-based or not.

Another way to classify CBE is by purpose or goal. For instance we can divide CBE into self-study applications and learning resources and collaborative learning environments and tools. Or we can divide the use of CBE into basic study material and testing, for example computer exams. In the case of basic study material, CBE allows the power of multimedia to be used for providing animations or perhaps audible examples together with text and pictures. Nevertheless, the goal is somewhat the same as for text books. In computer exams, the purpose is to evaluate learning regardless of the source used.

Naturally, we can also classify CBE-based on numerous technical criteria such as the software requirements etc. One of the most important technical categorizations is whether network, especially the Internet, is used or not and if so, to what extent. Furthermore, like all education, the Bloom's classic taxonomy [20] can be applied for CBE classifications as well!

2.2.3 Internet and Studying

The Internet and World Wide Web can be used for teaching and studying in many ways. For instance, the learners can use the web search engines for finding information and study material either for self-study purposes or as course tasks. Nowadays it is common that all courses have their own home pages with lecture material, instructions etc. Some universities have even decided to start putting all course material available on the web [68].

The development of interactive technologies for web has allowed various collaborative applications and learning environments. Moodle [32] and FLE3 [117] are examples of tools that emphasize the social aspects of learning. With the help of chat applications, message boards and other collaboration tools it is easy to form study groups even with long distances. Besides the above mentioned free tools there are also various commercial tools for building these learning environments, for example WebCT [132] and Hyperwave [52].

Another form of distance learning is making audio and video recordings of lectures and making them available on the Internet, either on-line or off-line. This kind of systems require special software tools such as media players and streaming applications. Video-conferencing has become quite popular though, not only for educational purposes but for general purpose meetings as well.

2.2.4 Self-studying

Self-studying is a special form of distance learning. There are two different ways to explore the Internet. Firstly, the learner may browse and find information when needed. With the help of fast and intelligent search engines, and on-line encyclopedia such as Wikipedia [9] it is fairly easy to find answers to questions. However, when thinking of learning, this approach has some limitations. For example, currently it is quite difficult to evaluate the reliability and origin of the information. Therefore, there is a risk of studying and learning wrong information. Also the easiness of jumping onto sidepaths may lead into losing of focus: after the study session the learner has learned quite a lot but not necessarily what he/she was looking after in the first place.

Another approach is to collect information onto ready-to-use packages. Browsing through a list of previously chosen resources makes the learning process more compact. A simple practical example would be a set of course pages or a link list.

2.2.5 Interactivity

Just browsing through static material on the web is not so far from reading and studying a book. At least multimedia demonstrations such as audio examples, images, video and animation should be used when appropriate to enhance the adaptation of material.

However, for many purposes, interactive content is desired. Interactivity means that the learner is able to affect and control the content in some way. A typical example would be a multiple-choice question or evaluation of simple mathematical equations. As presented later, acoustics, audio and digital signal processing are areas where interactivity plays a great role in deeper understanding of educational content.

Another viewpoint is that interactive multimedia content is the benefit of using CBE instead of regular group-oriented teaching methods such as lectures and exercises.

2.2.6 Usability

Usability is always one of the most important issues that should be taken into account when designing web applications or software in general. Educational applications are especially vulnerable to this. One have to remember that in human-computer interaction, the computer cannot use any of the additional, multimodal cues such as facial gestures or variations of speech that people constantly use in their everyday life. Therefore, under this limited interaction scenario, bad usability can easily ruin the whole learning process. Unfortunately, usability is not an exact science but very much application dependent [86]. Therefore, it is vital to recognize the importance of usability already in the design phase of educational applications and evaluate usability throughout the implementation process. Naturally, one should also familiarize oneself with some known principles and characteristics of good usability [85].

2.3 Evaluation of Learning

Without doubt, one of the most difficult educational questions is the evaluation of learning. Whether someone has actually learned something, and what might that be, is unique to each learner.

From the teacher's point of view, evaluation of learning aims to determine whether the students have achieved the goals set for a particular topic or course. Typically this is tested somehow and based on the test results, evaluation is performed. This kind of method is always restricted to the test, however. Many times the students may actually learn something completely different from those that are tested. Furthermore, testing does not necessarily take the students' individual needs into account. However, when properly designed, the testing method is a practical and even fairly reliable way of evaluating learning.

From student's point of view, taking a test is only a small part of the learning. For example during a course, the students may adapt and understand various things that are not key goals or even related to the core contents of the course. Therefore self-evaluation and self-reflection are important parts of learning. This requires metacognitive skills which may be achieved by experience and with the help of studying methods such as the learning diary, portfolio etc. Whereas the testing focuses on measuring whether certain goals have been reached or not, self-evaluation aims for deeper understanding of what and how one has actually learned.

Still, the testing method is important to students as well. The test results are an important feedback channel and have great effect on how students will go on with their studies.

2.3.1 "Semi-automatic" Evaluation

Computer-based education makes it possible to automatize, at least partly, the testing process. If the test questions are limited to simple, unambiguous answers, it is possible to directly evaluate the results. This kind of approach is fast and gives the students immediate feedback. However, the applicability of this kind of tests is somewhat questionable since in most cases, the test would be too limited with respect to the course or topic.

Another approach is to use computers for gathering the results and allow the teacher to evaluate the answers. This method allows wide selection of question types to be used in the test, but it only helps in getting the answers in a compact way. Otherwise, the method is similar to standard exams.

Somewhere in between is a "semi-automatic" approach, where software is used for gathering the results as well as partially evaluating them. This kind of approach is nowadays perhaps the most popular in the field of computer assisted testing and exams.

This thesis presents a method for semi-automatic evaluation of learning, which is based on monitoring and evaluating the users' path and actions when studying a self-study application. This method provides a view of the studying process thus giving also some insight on the learning process itself, not only the results. Additional information can be found in chapter 5.

Chapter 3

Acoustics and Audio Signal Processing

In this chapter acoustics and audio signal processing are presented as educational fields of interest in general and as content topics for CBE and web-based education. The goal is to provide background information on what kind of requirements they present to educational applications and what kind of tools are commonly used for educational purposes in these fields.

3.1 Special Requirements

Acoustics and audio signal processing are fields of special interest to education. First of all, it is obvious that the use of sound is involved in all education in these areas. Along with mathematics, figures and text, audible examples are needed. With the web platform, use of sound presents special technical challenges and requirements that are discussed in the later chapters [93].

Secondly, in any practical work related to these areas, digital signal processing (DSP) is used as a tool. Therefore, in universities all over the world various DSP and numerical tools are used not only for research work but also for education as well. In fact, for many university students, learning DSP or audio signal processing actually means writing code and trying it out in addition to standard lectures and textbooks. This means that also basic programming skills are needed.

3.1.1 Use of Sound

In order to teach and learn sound, one must use sound. This practical fact should be taken into account in all education of acoustics and audio signal processing. However, it also means that appropriate audio hardware has to be available. For most purposes, the modern computers have sufficient audio capabilities but adequate loudspeakers or headphones are also needed. In order to record signals, microphones are needed as well. For more advanced educational purposes, the quality aspects for these components should also be investigated.

3.1.2 Transferring Audio

In order to provide audio content over the network, one also needs to take transfer requirements into account. Similar to video and high quality images, audio also requires considerably more network bandwidth than for example text transfers. In fact, raw CD-quality audio requires $44100 * 16 * 2 = 1411200$ bit/s which can be reduced to approximately 1:7 - 1:10 by proper encoding. Furthermore, streaming technology solutions [91] can be applied for reducing the delay experienced by the user. Nowadays, the audio streaming technology has become established technology and is widely used for listening and purchasing music over the Internet. Even the home Internet connections have enough bandwidth to allow encoded audio data, for example MP3 music, to be downloaded with reasonable speed.

3.1.3 DSP

One of the most difficult technical problems with audio and digital signal processing is the requirement for computational power. In order to perform the actual processing, one needs enough computational power and tools. Even though the DSP operations basically involve only summations and multiplications, the amount of them as well as the calculation speed needed for practical applications is considerable. Therefore, despite that modern computers are very powerful and adequate also for DSP, complex algorithms easily become too demanding for real-time processing. Luckily, strict real-time requirement is not necessarily an issue for educational purposes: as long as the processing delay is tolerable in practice, it is enough.

For web applications the situation is more difficult. If there are many simultaneous users, the server-side DSP processing may eat up the server's computational power. Client-side technologies, however, usually have only a limited support for DSP operations and audio I/O due to computational and security limitations.

3.1.4 User Interfaces

Audio signal processing education also requires multimodal user interfaces. One should be able to play and control audio as well as easily access visualizations such as time and frequency domain presentations of signals. Furthermore, in many cases controlling and manipulating audio and graphical elements is needed: examples include simple volume adjustment as well as graphical select and zoom operations.

For more advanced level engineering education, one should be able to use mathematical expressions or algorithms as an input, not to mention audio as input. Especially in terms of the web interfaces, implementation of any of these is a challenging task.

3.2 Tools

The power of multimedia for DSP education has been a topic of interest for quite some time now. Throughout the years various software tools and packages have been developed in order to provide the students more "hands-on" experience with DSP. Out of these numerous tools, only a small set has been selected to be presented, categorized and discussed here.

3.2.1 Programming oriented tools

One of the most popular tools for DSP is Matlab® by Mathworks [69]. It is used not only for education but for research and development throughout the world. A wide selection of Toolboxes (programming libraries) makes it a general purpose environment for numerical analysis and mathematics oriented tasks. Several toolboxes include ready-to-use DSP functionality allowing rapid testing and simulation of algorithms. Matlab also includes support for GUI design and various multimedia file formats including audio, image, video and animation.

For educational purposes Matlab is suitable for programming exercises, simulations and interactive demonstrations. Quite a few DSP textbooks include Matlab exercises or software, for example [72], [88], [55], [67].

Other similar tools include Octave [35] and Mathematica® [133], the former being open source. All these tools require some programming skills and basic knowledge of their own programming languages.

3.2.2 Laboratory tools

Laboratory experiments are an important part of all engineering education. They give the students a chance to practice with special hardware and software adding a true "hands-on" feel to their studies. In a typical case the students perform predefined tasks in a laboratory equipped with appropriate hardware and software. Naturally, for acoustics and audio signal processing education, the laboratory space needs to be equipped with special audio hardware. Also practical acoustical problems such as noise need to be considered.

Examples of stand-alone laboratory-oriented educational applications include the psychoacoustical measurements [60] and the loudspeaker response measurements [115], both built with the QuickSig DSP environment [58]. A more general purpose tool for technical laboratory experiments is LabView® [81]. It includes both software and hardware to perform measurements and analyze the measured data. Matlab® [69] can also be used for similar purposes, provided that there are appropriate toolboxes and hardware.

3.2.3 On-line demonstrations

In this classification, on-line demonstrations refer to demonstrations that illustrate a single topic or only a few topics. Combined they may form a more complete educational package, or may be included in a larger educational context, but from the content point of view, their purpose is to demonstrate a single concept compared to larger goals of educational applications. In other words, even combined into larger collections, etc., they do not provide the contextual relationships between one and another.

An on-line demonstration can be simply audio files, images, animations or more advanced interactive demonstrations such as Java Applets or Flash applications, discussed more in chapter 4. They may differ in various respects such as educational goals, educational levels or implementation. In fact, the only common nominator is that they are available on the web.

An example of an on-line demonstration is a Java Applet allowing the user to interactively try out some simple digital filtering effects [96]. In this demo, the user can choose an audio sample and define some filtering parameters. After doing so, the audio sample is filtered and the user can investigate the processing as a time and frequency domain presentation as well as listen to the processed sample. The demonstration was implemented as a Java Applet, which is a client-side technology thus relying on the capabilities of the web browser.

Another audio-related example is the WWWVbap demonstration available on the Helsinki University of Technology Laboratory of Acoustics and Audio Signal Processing web pages [98]. The demonstration illustrates the VBAP method [90] used for multichannel audio reproduction. This demonstration is implemented using server-side technology only. Therefore, it can be operated with any web browser, but for the demo to be contentwise meaningful, the user should have an appropriate multichannel audio reproduction system to try it out.

In engineering education and research this kind of demonstrations are very popular and useful for both self-study and classroom purposes.

Virtual laboratories

As a special case of on-line demonstrations, the rapid growth of Internet has made virtual laboratories popular during the last few years. A virtual laboratory is a software, possibly a web-based simulation of a real laboratory. Typically they include a set of interactive on-line demonstrations that can be used for self-study education or to "dry runs" before actual laboratory experiments [48]. Some of these virtual laboratories may include on-line measurement data or even on-line control of laboratory equipment ("remote laboratory") [41].

3.2.4 Others

QuickSig [58] is a signal processing environment and tool written in the Common Lisp [101] language. It has been used also for creating stand-alone educational applications including hands-on signal processing experiments and hypermedia-like navigation.

"Introduction to Signal Processing" ([92], [60]) is an example of a stand-alone CBE application made with QuickSig. It included some tutorial level DSP material accompanied with interactive audio signal processing experiments. The practical setup included a computer and headphones located in the laboratory premises, otherwise there were no other requirements for placement or hardware. The application was also capable of keeping track of timing when students worked with it and producing a log file for later investigation. The CBE application was used in a course as additional material for a couple of years. A more detailed description of this application can be found in chapter 6.

Another example of a CBE application is a "Software Teacher for Loudspeaker Measurements" [115]. It is written using the same QuickSig environment. The stand-alone CBE application includes some basic theory of response measurements and related hardware and a real measurement. The theory part includes small questions that the users need to answer, followed by a measurement simulation and finally a real measurement. The practical setup involves constructing a real measurement into an anechoic chamber with real loudspeakers and microphones. In addition to provided equipment, the students are able to measure their own loudspeakers with the help of this application. The CBE application is actively used in a course at the Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing.

BlockCompiler [59] is a tool for advanced real-time modeling purposes. QuickSig is used as a higher level modeling platform but the actual DSP processing is done by creating C-code, compiling it on-the-fly and running the created "block" in realtime. The system can be used for especially for creating interactive demonstrations for educational and research purposes.

Pd (Puredata) [89], Max/MSP [29] and JMax [34] are visual programming tools that allow the user to create audio signal processing algorithms by using visual blocks and interactively control and run these patches. These tools can be used for various educational purposes. Being completely free and ported to various operating systems, Pd has become very popular in just a couple of years.

An interesting way to utilize Pd is through a browser plugin such as [44] and [5].The plugin allows Pd to be operated by using the web browser as an interface. The system requires that Pd, the browser plugin and some libraries are installed, but given those prerequisites, one can create advanced interactive demos and applications.

Finally, there are numerous code libraries and code examples available on the Web that can be utilized in education as well, for instance [6] and [70].

Chapter 4

Web Technologies

In this chapter, the principles of web technologies are introduced and explained in detail. The purpose of this is to give technological insight of the web-based education also to readers who might not be familiar with application development for the web and to provide necessary background information for anyone interested in developing their own web applications. Furthermore, the chapter tries to show why the web is not necessarily the optimal platform for applications regarding high level management of interactivity and users.

Please note that the HTML code in all the examples is simplified in order to demonstrate the programming technique in question and is not valid HTML for real life use.

4.1 Web Architecture

The web (World Wide Web, WWW), like Internet services in general, is based on a client-server architecture (see Figure 4.1). The client (i.e. the browser) sends a request to the server using the HTTP protocol [40]. The server processes the request and returns a response using the same protocol. Typically the response is HTML [123] [121] data, but it can be some other type of multimedia content like images, audio etc. In fact, the HTTP protocol can be used for transferring various forms of data.

However, the key nature of this transaction process is that it is strictly request-based. All requests are served one at a time and therefore, each request is unique and has no history. The basic web architecture does not support any kind of grouping or evaluation of multiple requests. This has a strong impact on implementation of interactive services and, especially, produces various security issues.

Another fundamental characteristic of the web architecture is that it has no real support for interactivity. The HTTP protocol is capable of transferring data in two directions and it has built-in features for simple transfer of textual parameters, but other than those, any

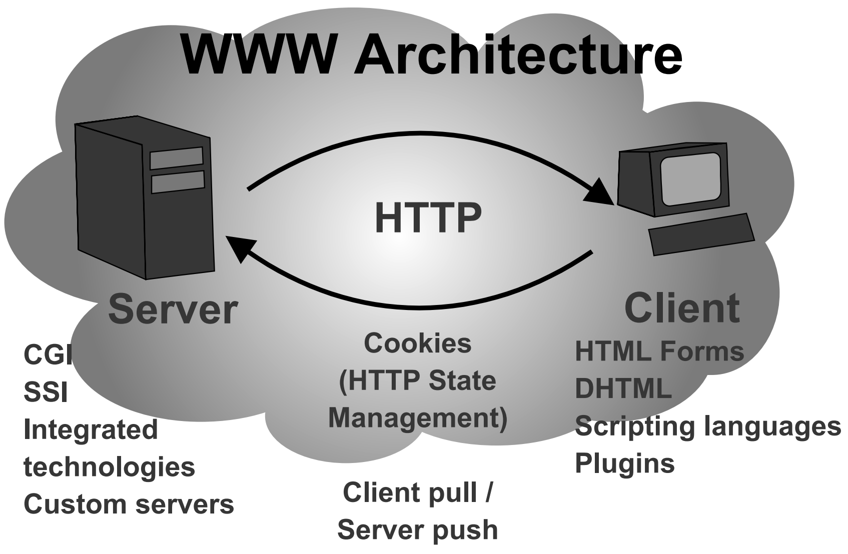


Figure 4.1: Basic web architecture

interactivity relies on the capabilities of browsers and servers. To overcome this limitation, numerous server and client-side technologies have been developed, some of which have indeed become de-facto standards.

4.1.1 URL /URI

Unified Resource Locator (URL) and Unified Resource Identificator (URI) [16] are addressing standards that describe resources on the web. In standard web operation, the browser sends a request containing the resource URI to the server. The server then interprets the URI and provides appropriate response. In a simple case, the URI defines a path to the desired HTML file. In many cases the URIs do not actually refer to any distinct file but to a program creating the desired response on-the-fly utilizing databases etc. On the other hand, the URI is an important tool for finding information, especially by means of saving the addresses for later use.

4.2 Methods for Interactivity

Interactivity means that the user has a way of giving input-based on which the response is created dynamically. A simple example is a search engine: first the user types in search words and sends it to the server. The server does a database lookup and returns matching data (note, however, that the internal structure of the search engine can be very complex, think about for example Google, http://www.google.com). For users, this is a simple and natural way of finding information. However, things get more complicated when the user would like to choose items graphically or continue the search process by performing additional searches etc. The method of describing an interactive process only as textual input and request-based operation, is very limited and not sufficient for advanced applications.

Due to the lack of underlying support for interactivity in the basic web architecture, a number of methods have been developed to overcome this shortage. These can be categorized in many ways, but perhaps the most fundamental distinction is the one between client and server side processing (see Figure 4.1) discussed in this chapter. With server-side technologies, the server produces the desired output based on parameters given at the request. In client-side technologies the situation is the opposite, the browser receives programmatical instructions and processes them. For many practical applications, a hybrid is needed: both server-side and client-side technologies are used.

Further divisions can be made based on, for example, how the program code is processed or located with respect to normal HTML data.

4.2.1 Server-side Technologies

The principle of operation for server-side technologies is that, rather than simply reading contents of predefined files and sending them over to the browser, the web server produces the content dynamically. Therefore, instead of having to create all the web content beforehand, web authors can write program code to handle varying requests. For interactive services, the requests may include input parameters given by the user, but this is not always the case. Server-side technologies can also be used for producing dynamic content without any user input. All and all, server-side technologies make it possible to control how resources are used and distributed in complex program structures. Therefore, it is no surprise that these technologies are widely used for almost all kinds of web sites.

One of the major benefits of the server-side approach is the independency from various browser capabilities other than HTML. The browsers need not to have any special features besides regular viewing and requesting web content. Furthermore, the server-side operation allows content providers to fully control the content.

Common Gateway Interface (CGI)

The first and still widely used server-side technology is the CGI (Common Gateway Interface) [82]. It defines how and what kind of parameters the web server transfers to a program ("CGI script"). It also defines how this program is called, but this does not affect the interaction process itself. It does have a substantial influence on the server performance, though.

The CGI programs can be written in any programming language, but some of them are more suitable to this purpose than others. C and Perl are perhaps the most popular ones. For well known security reasons, shell scripts should be avoided. The CGI specification is also server-independent and supported by most web servers.

From the client-side, the CGI program expects parameters. Typically, these are given in HTML forms [121]. The HTML specification includes a number of input-type elements which can be used for giving textual input. These parameters are then delivered to a CGI program using special environment variables. Note, however, that the HTML input elements can be used together with any server-side technology, not only CGI, even though they originally appeared in the HTML specification together with the development of the CGI. In fact, the HTML forms are the basic user interface elements for delivering parameters to the server regardless of the technology used for handling these parameters.

A special type of input is the file upload mechanism [83]. Instead of sending just textual parameters, the user can specify a file that the browser uploads to the server. With CGI the file is handled through standard input and some additional parameters are given specified in environment variables. The file itself may contain whatever kind of data (text, binary etc.), it is up to the CGI program to decide what to do with it.

After running the CGI program, the web server captures its output and sends it back to the browser.

It was soon discovered that the CGI is not very efficient in its way of using external programs to process data. Also the method for providing parameters in environment variables is a little clumsy, although it does serve the purpose. Furthermore, the CGI programs are somewhat limited in terms of accessing the server internal or other resources.

Server modules

Apache [10], one of the most widely used server software, has a modular architecture. It allows additional modules to be created to enhance the core server functionality. Examples of such modules are mod_perl [11] and mod_php [113], which tie Perl [131] and PHP [65] run-time environments closely into the server. This reduces the server load dramatically with the expense of increased memory usage though. These modules also allow new methods for accessing parameters (for example with PHP, the input parameters can be automatically associated with program variables) and also allow direct processing of program code within HTML documents. This makes it possible to divide the program into smaller segments, components, inside a regular HTML page. Currently there are various

Apache modules available for different purposes. Even the core operation is actually based on modules.

Other server integration solutions

Other examples of server-side integrated solutions are Java Servlets and Java Server Pages (JSP) [50], originally developed by Sun for the Java Web Server product [110]. Nowadays there are various implementations for a number of server products including a module for Apache. Being Java-based technologies, JSP and Servlets require the Java Runtime Environment (JRE) [108]. However, they also allow the use of numerous Java libraries and programming APIs having made them popular and flexible solutions for many complex web services.

Microsoft has also developed their own solution called Active Server Pages (ASP) [79]. This is similar to the others mentioned except that it is available for Windows server platforms. There are also server specific custom programming languages, for example Lasso [87]. All of these have their pros and cons from the developers point of view.

Application servers

The application server concept introduces a higher level solution for building web applications. An application server provides ready-to-use components that can be used as building blocks for web services. Typically, some database functionality is also included in the server. This way the needed services can be easily built and customized for different purposes and content providers can buy or rent those components. Many web stores utilize this kind of services for example with the processing of credit card transactions etc.

Technically, application servers are web servers with integrated functionality for frequently needed web programming tasks and customization. Typically, they utilize XML [119] and/or SQL [62] databases for inner representation of data and provide programming libraries or proprietary programming languages for developing applications. The J2EE framework [105] seems to be quite popular in terms of implementations.

Later Content Management Systems (CMS) will be discussed. In the last couple of years, these have became popular especially for small-scale non-commercial web service development.

Specialized servers

Because the basic HTTP functionality is fairly easy to implement, a web interface has been built into a number of other server software. A typical example would be a modern printer which can be configured and monitored using a web browser over the Internet. Typically these kind of solutions cannot be used as general purpose web servers but only introduce functionality for limited tasks.

Another type of specialized server is a server that has been programmed to provide only a certain functionality. They are usually very efficient but of course, content cannot be easily added or modified. From the point of view of acoustics and digital signal processing, these might be of interest for example in tasks such as audio streaming, providing DSP computation etc.

Complete web programming frameworks

J2EE [105] and .NET [80] are programming frameworks for developing web applications and services. These frameworks provide extensive libraries, programming APIs, programming tools etc. for complex, distributed applications and content presentations. For simple tasks and web applications these might be too heavy as such, but especially for commercially targeted and complex distributed web services, these frameworks provide many useful solutions and programming practices. For J2EE the programming language is Java, for .NET C# (pronounced C sharp).

4.2.2 Server-side programming techniques

The various server-side technologies have also affected the programming techniques. These techniques typically aim for one goal which is to separate the content and the programmatical logic. If successfully implemented, this separation allows the appearance of web pages to be changed and designed separately from the programmatical logic and vice versa. In practice, the separation cannot be fully implemented because in many cases, the changes are not only cosmetic but affect both the appearance and the programmatical logic of the application. Such separation can also be questioned philosophically, but nevertheless, the goal for many web programming techniques is to technically separate the HTML design and the program logic. Later on, Cascading Style Sheets (CSS) [18] have provided additional means for controlling the appearance of the web pages.

The distinction between the program logic and the HTML design is emphasized by the fact that many different browsers are used for accessing the services. Especially many older browsers do not support all the latest HTML or CSS specifications. As a matter of fact, many browsers accessing the web material are not actually interactive browsers but search robots and spiders. This makes the technical use of HTML and CSS a challenging task and further emphasizes the need for separating the content and its appearance. The concept of machine-understandable data on the Web is further investigated by the W3C Semantic Web working groups [122], [17].

Templating

The very first CGI scripts produced HTML code directly as such. It was soon realized that this is not very convenient if changes were needed. Therefore, a natural solution was to use a template instead of hard coding all HTML into the script. The template is a special HTML file that has markings for programmatical content. When the script is called, the template is read in, the markings are replaced with program results and finally, the constructed page is returned to the user.

This way the templates provide an easy and natural way to separate the page design and the programmatical content such as a database query etc. Furthermore, it allows easy modification of the page layout, because it means only editing of the corresponding template file.

A practical "bonus" for using templates is that the template files do not need to be placed available for the public web area. Instead, only the script needs to be accessed. This can also be considered a drawback: the output of the script depends on 1 to N template files which may or may not be accessible at the time.

Here is an example of the templating technique using Perl and the HTML::Template-module [116].

The contents of the template file (template.tmpl):

```
<html>
 <head>
 <title>Test Template</title>
 </head>
  <body>
  Today is <TMPL_VAR NAME=DATE>
  <p>
  The time is now <TMPL_VAR NAME=TIME>
 </body>
</html>
```

A simple CGI script (template-example.cgi):

```
#!/usr/bin/perl -w
  use HTML::Template;

  # open the html template
  my $template = HTML::Template->new(filename => 'template.tmpl');

  # fill in some parameters
  my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
    localtime(time);
  my $date = $mday . "." . ($mon + 1) . "." . ($year + 1900);
  my $time = $hour . ":" . $min . ":" . $sec;
  $template->param(DATE => $date);
  $template->param(TIME => $time);

  # send the obligatory Content-Type and print the template output
  print "Content-Type: text/html\n\n", $template->output;
```

Output of the script when requested by a browser (http://someserver/path/template-example.cgi):

```
Today is 29.11.2005
The time is now 14:36:01
```

Embedding code into HTML

As a contrast to the templating techniques, PHP programming language was perhaps the first to include the program code directly into the HTML files. This allows a similar separation of HTML and program logic, but instead of having just markings on the template, code snippets can be used. When requested, the server reads the code, evaluates their results and joins them with the rest of the content. This method logically separates the HTML and the program code, but still keeps them in the same file. Furthermore, the embedding method keeps the logical connection between the file and its URL.

PHP, JSP and ASP are all based on embedding and there are several solutions for other languages as well, Perl in particular. The following PHP example clarifies the embedding method in practice.

The contents of the PHP file (embedded.php):

```
<html>
 <head>
 <title>Test Template</title>
 </head>
  <body>
  Today is <?php echo date("d.m.Y");?>
  <p>
  The time is now <?php echo date("H:i:s");?>
 </body>
</html>
```

Output of the script when requested by a browser is similar to the previous example (http://someserver/path/embedded.php):

```
Today is 29.11.2005
The time is now 14:36:01
```

Components

In the template method, the HTML code contained special markers to include programmatical content and with embedding, these markers were replaced by program blocks. With both methods either the markers or the program blocks can be thought also as calls to separate functional blocks. This is the idea of components. Rather than controlling the overall page contents all the time, the program logic is divided into smaller functional components. For example, there might be a component that provides a navigation bar to each page. When the site structure changes, it is handy to have each page call a component, a subroutine etc. to provide the navigation bar instead of updating the code in all pages. Typically, the components are packed into a library which is then used in all scripts/pages.

Even more advanced solutions include higher level programming languages and APIs for constructing the pages from skeletons containing only calls to components. Mason [111] is an example of such advanced system written in Perl.

In practice, the distinction between these techniques is somewhat blurry. Many sites use templates, embedding and components regardless of the actual programming language and even multiple techniques together. The following PHP example clarifies the components in practice.

The contents of the PHP file (component-library.php):

```
<?php
function htmlheader() {
echo "
 <html>
 <head>
 <title>Test Template</title>
 </head>
 <body>";
}

function printtime() {
echo "Today is " .  date("d.m.Y");
echo "  <p>"
echo "The time is now " . date("H:i:s");
}

function htmlfooter() {
echo "
 </body>
</html>";
}
?>
```

The contents of the PHP file (components.php):

```
<?php
require("component-library.php");
htmlheader();
printtime();
htmlfooter();
?>
```

Output of the script when requested by a browser is similar to the previous examples (http://someserver/path/components.php):

```
Today is 29.11.2005
The time is now 14:36:01
```

4.2.3 Client-side Technologies

Another approach for developing interactive applications is to use client-side technologies. The idea is that program code is sent to the browser, which then evaluates and runs the code. Typical examples of this kind of solutions are Javascript [49] and Java Applets [106]. With the first, source code is sent to the browser and with the latter, compiled bytecode is used and the browser starts a virtual machine to run the code.

In terms of security, this kind of approach is problematic. Downloading program code from an unknown source and running that code involves major security concerns for the user. Therefore with most client-side technologies, the functionality is restricted or special security mechanisms have been built into the solutions.

Another major consideration with client-side technologies is that they are all browser-specific. Whether the code runs successfully or does what it was intended to do, depends on the capabilities of the browser used. For application developers and service providers, this is a major concern, especially with usability issues.

Javascript

Perhaps the most common client-side technology is Javascript, which is also one of the oldest. Today, there are several versions and variations available. With newer browsers, the most widely accepted specification is called ECMAscript [36] which is based on Javascript version 1.3. It attempts to reach stability and overcome the differences between browsers. ECMAscript is not a new language, only a specification for Javascript. That is why the name Javascript is commonly used even if the language used would be ECMAscript conformant.

Through the Document Object Model (DOM) [120], Javascript provides methods for accessing different parts of the HTML document and modifying them. We have to remember that basic HTML does not provide us any means for interaction. Therefore Javascript is a natural choice for simple user interface tasks. However, Javascript is somewhat limited in terms of computational capabilities and, furthermore, the operation is strictly limited to the browser to ensure safety of the user.

Javascript, DOM and Cascading Style Sheets together form a technology commonly known as Dynamic HTML (DHTML). Unlike it's components, DHTML is not a specification but rather just a practical method for developing user interfaces with web browsers.

From security point of view, one should realize that Javascript source code is always visible to the user. Furthermore, together with web servers that dynamically generate web pages based on unvalidated input, Javascript can be used for so called cross site scripting attacks [27].

The date and time example written in Javascript using two different styles (javascript-example.html):

```
<html>
 <head>
 <title>Test Template</title>
 <script type="text/javascript">
var now=new Date();
function currentdate() {
document.writeln(now.getDate() + "." +
(now.getMonth() + 1) + "." + now.getFullYear());
}
 </script>
 </head>
 <body>
  Today is <script type="text/javascript">currentdate();</script>
  <p>
  The time is now <script type="text/javascript">document.writeln(
  now.getHours()  + ":" + now.getMinutes()  + ":" + now.getSeconds());
</script>
 </body>
</html>
```

Java Applets

Java Applets, also one of the oldest client-side technologies, gives developers more control over the appearance and application behavior. With Applets, compiled bytecode is downloaded after which the Java Virtual Machine (JVM) is loaded and runs the code. In the HTML document, special tags are used to define a certain space that the Applet may use [106].

Securitywise, Applets operate in a sandbox. With this model, the Applets are allowed to do pretty much anything inside the sandbox, but have only a restricted access to the outside world. This makes them quite secure, but also presents problems with certain operations such as accessing the audio hardware in the local computer. Particularly, an Applet running with the Applet security manager can play, but not record, audio [109]. However, the user may configure his/her local Java environment to allow recording of audio with Applets.

Despite being one of the oldest and well documented client-side technologies, the Applets have suffered from unstability between browsers. Both the available Java Virtual Machines and the methods for using them in different browsers have been changing quite often, resulting in browser crashes when loading Applets etc. Recently a study and testing application was created to test Java Applets in practice. The results of this study were not too promising, in fact from the developers point of view, achieving a stable, browser independent Applet seems somewhat impossible in practice [15].

The date and time example written as a Java Applet (DateExample.html):

```
<html>
 <head>
 <title>Test Template</title>
 </head>
 <body>
 <applet code="DateExample.class" width=150 height=25>
 </body>
</html>
```

And the java source file for compiling the class-file (DateExample.java):

```
import java.applet.Applet;
import java.awt.Graphics;
import java.util.*;
import java.text.*;

public class DateExample extends Applet {
    public void paint(Graphics g) {
        Date now = new Date();

        DateFormat df = DateFormat.getDateInstance(DateFormat.SHORT);
        String datestr = df.format(now);

        DateFormat tf = DateFormat.getTimeInstance(DateFormat.SHORT);
        String timestr = tf.format(now);

        g.drawString("Today is " + datestr, 10, 10);
        g.drawString("The time is now " + timestr, 10, 25);
    }
}
```

Plugins

Perhaps the first client-side technology was the plugins [76]. The idea originated in a fact that including all necessary functionality for handling various multimedia elements would result in huge and heavy browsers. The plugins API [74] allows small software components to be used for interacting with the operating system and other software. Examples of such plugins are the QuickTime plugin [13] which provides a player for various audio and video formats and the PDF plugin [1] (Portable Document Format) which interacts with Acrobat software in order to view PDF documents directly with the browser. Also Java Applets are often used through a plugin [107].

A popular and interesting plugin is the Macromedia Flash plugin [2], which allows Flash applications to be included in the web content. The plugin is freely available for a number of platforms. Flash applications have good support for many multimedia elements such as audio, video and images. Therefore it has become quite popular for building advanced user interfaces, small games etc. The technology is also quite robust which has made Flash replace Java Applets in many cases.

The plugins are platform dependent and problematic securitywise. The plugins API does not provide any security whatsoever and plugins may use any operating system resources

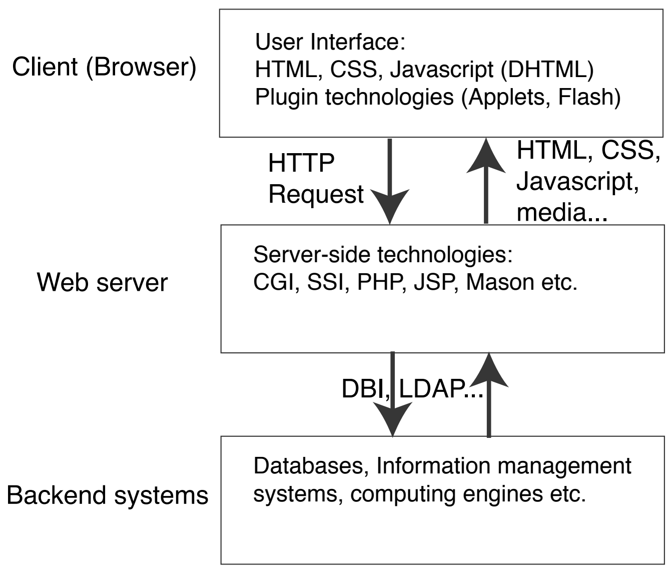

Client (Browser)
User Interface:
HTML, CSS, Javascript (DHTML)
Plugin technologies (Applets, Flash)
HTTP Request
HTML, CSS, Javascript, media...
Web server
Server-side technologies:
CGI, SSI, PHP, JSP, Mason etc.
DBI, LDAP...
Backend systems
Databases, Information management systems, computing engines etc.

Figure 4.2: Hybrid web applications

freely.

Modular browsers

One of the more recent approaches was taken by the developers of open-source Mozilla and Firefox browsers. These browsers have a modular architecture making it possible to implement features as extensions [75]. This way the core operation of the browser can be kept fairly lightweight and the browser can be customized for individual needs. For this, a special user interface language called XUL [77] was developed and included in the core browser functionality. Unlike plugins, the extensions are by default platform independent thus making them an interesting alternative for adding advanced functionality to the browser.

4.2.4 Hybrid Models

In practice, most web services nowadays use hybrid technologies, i.e. both server- and client-side technologies. This is a technical reality because for advanced applications neither approach alone provides the tools necessary for implementing the desired interactivity. Typically, Javascript and DHTML are used for building the graphical user interface whereas core application logic is built with server-side technologies. For more advanced user interfaces, Java Applets or Flash may be used. Figure 4.2 presents a typical web application architecture.

Ajax [46] is a more advanced technology for building web applications. It extends the classic hybrid model by XML-technologies and, especially, asynchronous interaction pattern by using its own Ajax engine for both rendering the user interface as well as communicating with the server. Implementation requires some technologies (particularly XMLHttpRequest [73]) only available in the recent browsers but no additional software like plugins, extensions etc.

4.2.5 Content Management Systems

In the last few years an interesting field of web applications has arisen to help the growing need of various interactive and community services. These are called Content Management Systems (CMS) and they are typically based on free and/or open source web technologies and products such as PHP, Java, SQL databases and Apache. The basic idea is that the CMS systems offer users tools for updating the content. This way it is easy to distribute the webmaster duties and keep sites up-to-date.

There are numerous CMS software available ([7], [112]) varying in complexity, features and requirements. Many of them are open source projects themselves and have evolved from a simple set of scripts to huge database driven systems with their own component programming and templating languages. Some of these systems are targeted for group and community services, some for eCommerce, some for WiKi systems [8], for instance Wikipedia [9]. Examples of education oriented open source CMS systems are Moodle [32] and Fle3 [117], which both focus on Computer Supported Collaborative Learning (CSCL). Like discussed earlier, Moodle emphasizes the pedagogical concept of constructionism, i.e. that making it visible enhances the learning process. Fle3 on the other hand emphasizes progressive inquiry learning. Also a wide variety of commercial content management systems for educational purposes, i.e. eLearning tools are available, for instance WebCT [132], Hyperwave [52] etc.

For many practical purposes the CMS systems are an alternative well worth investigating. However, one usually needs to study the CMS specific features, requirements and operation before making decisions, which might be a time-consuming task. Luckily, there are web sites, which present product comparison tools and example installations, for example the CMSMatrix [7] and Open Source CMS [112].

4.3 Control

Due to the nature of the web being request-based, one of the key issues for many advanced web applications is to how to identify and control a set of requests. Also managing users is essential in order to control access to resources.

4.3.1 User Management

User management stands for methods used and needed for distinguishing various users from each other. Typically user management involves a registration service and authentication of users. There are a number of ways to implement the authentication, perhaps the oldest and most popular being HTTP Basic Authentication [43]. With this method, the user is asked for a username and password, which are then kept in browsers memory and added to every HTTP request that is addressed to the service that asked for them in the first place. This method is easy to use and implement but it has a few drawbacks. First of all, the method is insecure in a way that the username and password information is frequently sent over the network without any encryption. Secondly, the browser keeps the username and password in memory until explicitly asked to forget them or the browser is closed. This means that with this authentication method, there is no control over time once the authentication is successfully accomplished.

The first problem, plain text passwords, can be overcome by using a so called secure server. A secure server is a regular web server but the browser and server first change keys that are used for encrypting all traffic between them. This technology is known as Transfer Layer Security [19] and it can be used with a number of other protocols other than HTTP as well. For web services that use basic authentication or otherwise sensitive information (such as personal details, credit card numbers etc.) it is recommended that a secure server is used instead of a regular web server.

Even the simplest user management schema involves of course also a database of valid usernames and passwords. A more advanced solution would be to use a separate authentication service. How well such a service can be integrated into web services or web applications varies in practice. Nevertheless, from both usability and security points of view, the authentication process should be done only once per session. This is simply a matter of implementation: there is no need to exchange authentication data all the time, a better solution is to authenticate once and let the rest be taken care by session control, discussed next.

4.3.2 Session Control

One of the most challenging areas of web programming is the session control. A session is a set of requests and responses that together form a group. A web application typically uses one or more sessions. Session control stands for methods needed to start and end a session, identify requests, keep track of the session, calculate unique ids etc. Session control is needed for content adaptation, personalization and reliable processing of state-based applications. Session control always requires server-side technologies to be used.

PATH_INFO

The oldest solution to implement session control was set in the CGI specification: the PATH_INFO-environment variable [82]. This variable can be given a unique value which is then attached to all requests as part of the URL. However, the id is visible to the users as well, which makes it unreliable because the user may easily modify the id value. In the worst case with too simple ids, the user may even gain access to a session by another user. Still, the PATH_INFO-variable is a useful tool for session control, provided that the unique ids are really unique and difficult enough for users to guess. Typically, the unique id is formed using the IP address of the browser sending the request, time and random numbers. With this method, like typically in all web programming, the possibility of users modifying the id's has to be taken into consideration beforehand.

Cookies

Perhaps the most popular single method for session control is to use HTTP cookies [64]. A cookie is a small textual information that is set by the server, stored to a disk by the browser and attached to all requests addressed to the service that set it in the first place. Cookies may have a time limit and valid address space, but they are also limited in size. Similar to HTTP Basic Authentication, cookies are transparent to the user and travel in the HTTP headers. This makes them a practical solution. With session control, a unique id is given as a cookie and then read back with each request to identify requests. Thus the cookie identifies the browser unambiguously.

Cookies have some unfortunate side effects and can be misused by service providers. The life time of a cookie can be set freely, allowing them to be set for a long time. This makes it possible for service providers to use cookies for tracking users, without users even knowing about it [47]. This has given cookies a very bad name, even though the cookies are harmless as such. Therefore, for session control purposes, the lifetime of the cookies should be set to current session only. Modern browsers also include advanced control of cookies that may make the cookies completely unusable to web developers.

Other methods

Other methods for session control include the use of the HTTP_REFERER environment variable [82] and hidden form fields. The former can be used for determining what was the referring page and the latter can be used for complete navigation provided that one script handles all the requests.

IBM developed a so called reliable HTTP (HTTPR) [114] to solve the session control and other problems related to stateless HTTP. However, HTTPR never received sufficient industry agreement around it. Therefore, IBM has continued with a new protocol, Web Services Reliable Messaging (WS-RM) [53] together with some industry partners.

Discussion

All of the above methods are somehow unreliable or suffer from limitations. That is why implementation of a reliable session control requires use of many methods combined. First of all, the PATH_INFO-method together with cookies ensures a fairly good control. Even better way is to use two unique ids instead of just one. This way they can be compared and proceed only if they both match, However, the crucial element in these methods is in fact the unique id used. To ensure reliability, the unique ids have to be long enough and random enough so that they cannot be easily guessed. For example the Apache web server has a module (mod_unique_id) [12] that generates a unique id for each request using various methods for making it random, yet safely alphanumeric. Secondly, the use of authentication and secure server with strong encryption ensures that the information cannot be modified easily. Modern Internet banking services utilize all of these techniques to ensure reliable operation.

One interesting technical problem related to session control is what information should be saved and how to store it. For storing the information, typical solutions are temporary files and/or database. In both cases, the information is stored so that it can be accessed with unique id. Also to prevent denial-of-service attacks one has to use time-based control for the session information and regularly clean up old session information. Otherwise the server disks may become full or database capacity limits reached resulting in a non-functional service.

What information should be stored, depends on the application itself. For some cases it is enough to store only the current state of the web application, for example the page that the user is currently exploring. Sometimes the state itself can be quite complicated making it difficult to design the session data structure and verification of the data.

However, many applications require also earlier information, i.e. that the history of usage of the application is stored as well as the current state. This is the case for example for applications using session control for content adaptation.

4.4 Logs

All general purpose web servers keep log files of the requests that they have received and processed. This is necessary so that the server resources such as disk capacity, network bandwidth, computational power etc can be efficiently used and prevent problem situations. Furthermore, in terms of security, the logs are essential in detecting and solving hacking attempts etc. [94]

A typical web server stores at least the date and time, IP address and the request into the log file. Additionally, the browser (User Agent) and referring page (HTTP_REFERER) might be stored. For busy servers, the log files easily become large and special log analyzer software is needed for parsing the files efficiently. If session control is used, session identifiers can also be stored in log files. When using any server side technologies, information similar to log files is accessible to each request. However, parsing log files directly during processing of requests is both inefficient and impractical.

It should be noted that the log files are of special interest also with privacy concerns. Even though the information stored in log files does not reveal any privacy information as such, they might be useful for finding such later, especially the IP address. However, the privacy concerns should always be taken into account when developing applications, especially if also user authentication is used.

4.5 Content Adaptation

Content adaptation means that each request and response are adapted based on user's earlier choices. A simple example would be a narrowed down search and another a survey, where upcoming choices are determined based on answers in earlier questions. In this field, known as adaptive hypermedia (AH) [25], active research is being done on adaptation algorithms and user modeling techniques [24] as well as systems and frameworks such as AHA! [22].

The AHA! [3] is an open-source, Java-based adaptive hypermedia engine, which has been developed at the Eindhoven University of Technology, in the Database and Hypermedia group. AHA! provides content adaptation by conditionally including fragments or objects and through link adaptation by conditionally changing the color of link anchors. The adaptation is performed each time a page is requested from the server. Furthermore, AHA also includes possibilities for the adaptation of the presentation of the content and even a possibility for the end users to change their user model.

The adaptation in AHA! is a two-stage process. Authors can define rules that translate user interactions (page accesses) into user model updates. The user model instance is then used to determine how the presentation is adapted. There are three ways to author the adaptation rules: by generating the rules from concept relationships using the graph author tool, by manually creating the rules using the concept editor tool or by manually editing an XML file containing the concepts and adaptation rules.

However, even though the AHA! system is a sophisticated tool for implementing adaptive web content, it has not been designed for educational applications particularly in mind. For educational applications, there is a possibility to present and evaluate adaptive multiple-choice tests, which can even be randomized to avoid circulation of lists of answers among students. However, the user interaction is still somewhat limited to simple link following or multiple choice questions, which limit the applicability of the system for educational purposes.

Further discussion of content adaptation can be found in the later chapters.

4.6 Personalization

Personalization can be understood as a certain type of content adaptation, but the term personalization usually refers to allowing users explicitly set some preferences with respect to the application [21]. The preferences can include layout related options such as "plain HTML" vs. "Flash content" etc. This kind of preferences are sometimes critical because of limited bandwidth or browser capabilities. Ketamo [61] has studied this kind of user and platform adaptation in web-based learning environments.

Especially in the case of users with disabilities, personalization is necessary. For example a blind person needs to browse the web with the help of a speech synthesizer and "fancy animations" etc. serve no purpose for the user. The World Wide Web Consortium (W3C) has released special accessibility guidelines for developers and service providers [118].

4.7 Agents and Distributed Computing

An agent is a piece of software that operates between the user and resources. The classic definition says that the agent has to be independent and aware of itself [39]. For this work, however, the concept of agent is understood in a simpler manner: it is simply a software construct that represents and models both the user and the teacher.

Furthermore, this thesis does not focus on multi-agent systems [39], currently an active research topic in the field of distributed computing and information systems. Examples of such topics include the Semantic Web [122] and the DARPA Agent Markup Language (DAML) [30]. Basically these define specifications for presenting data and metadata in a way that would allow computers to understand information and deal with information relations. This would allow much more advanced searches, a cornerstone for the future web infrastructure.

Distributed computing is also an interesting topic regarding this work. Basically it means accessing computational resources over the network. Together with the methods for accessing information, also methods for calling software routines in other computers have been developed as well as methods for defining web services in a way allowing programmatical access to the services. Specifications such as SOAP/XMLP [130] and XML-RPC [99] are examples of the former. The Web Services Description Language (WSDL) [124] is a recommendation for describing an interface to a web service such as an internet store, a search engine etc.

4.7.1 Learning Metadata

Even if metadata would be technically defined in a standard way like XML and RDF [126], the metadata content is always application specific. In order to exchange information, various metadata standards have been developed. These define what information should be included in the metadata. From educational applications point of view at least the IMS Learning Resource Metadata [56], Dublin Core Metadata [33], IEEE Standard for Learning Object Metadata [54] specifications as well as information collectives such as Ariadne [14] and Globe [37] are of interest.

4.8 Security Considerations

When developing any Internet service, security is always an issue. Interactive web applications are no exception to this.

The golden rule for any web programmer is that because of the stateless nature of the web, each request has to be verified every time. No information that comes from the browser can be trusted at any stage of processing, even if session control techniques were used for identifying requests. For advanced web applications, this verification process might become one of the key technical challenges in implementation because they involve a number of layered technologies. Further discussion can be found for example in the W3C Security resources [103] and Lincoln D. Stein's book [102].

Furthermore, no information received from the user should be passed on unfiltered even if it would not be needed for the actual processing on the server side. There are various examples of so called cross-site scripting attacks [27] where unwanted scripting information have been passed through a badly written web service.

Chapter 5

Proposal for a System Architecture

The previous chapters have presented and discussed the educational and technical background for interactive web-based CBE applications. Based on that background, there is a need for a system level architecture that takes the special requirements of educational applications into account. This chapter presents a proposal for such system level architecture, originally presented in [95]. The architecture tries to address the needs for interaction and adaptation required by educational applications and, even though the proposed architecture itself is abstract and general purpose, its origins and motivation lie in the fields of acoustics and audio signal processing. Furthermore, the question of semi-automatic evaluation of learning using this architecture is discussed.

Even though the solution proposal itself is not application- or platform-dependent, such an architecture can be implemented in numerous ways. The most appropriate methods of implementation do depend on the desired application or applications but also on available resources in terms of server software and hardware, amount of users, level of computational load etc. More discussion can be found in the next chapter.

5.1 Requirements and Design Goals

The most important requirements and goals for the design of the architecture have been

- Session control: Capability of identifying the user and his/her activities during a session
- Control of interactivity
- Independence of platforms and applications
- Support for multiple simultaneous users
- Support for multiple CBE applications

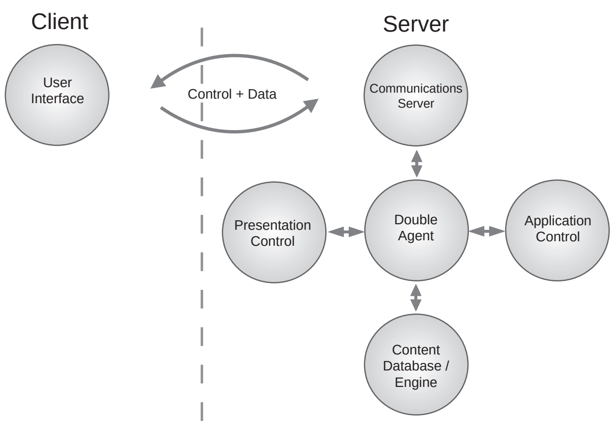


Figure 5.1: An overview of the architecture

- Compliance with existing technologies
- Extendability to future technologies

Many of these design principles are common to any advanced software design or Internet services. Therefore, it is obvious that the architecture can be used for purposes other than education as well. Still, the primary background and design goal has been to provide a system suitable for self-study CBE applications.

5.2 The Double-Agent Architecture

The solution, a double-agent architecture tries to match the above mentioned criteria. It is based on a software agent which is responsible for both analyzing users' requests as well as providing appropriate content, hence the name "Double-Agent architecture".

5.2.1 Functional Description

A functional description of the architecture is presented in Figure 5.1. From a high-level point of view the architecture consists of the user interface, the double agent and content. The essential idea is that every request made by the user is verified and processed by the agent first. Therefore, the user can only see the agent and all functionality is hidden behind the agent. Furthermore, the agent is responsible for all responses sent back to the user. Therefore, the agent also has control over what the user sees. This dual nature of the agent gave the name “double agent” and basically simulates an educational process similar to the one between a student and a teacher.

However, behind the scenes, plenty of details are needed. First of all, the agent needs a session control mechanism to identify and verify the request and evaluate it with respect to earlier requests. A basic session control functionality requires authentication, request identification mechanisms, request history and logging. Attention is needed also for certain computer security considerations. As discussed before, there are several ways to implement session control.

Some additional parameters related to user's personal preferences or available technical environment are also needed. For example, the user's browser software may only support certain features or the user may want to set some accessibility options. A special case of interest to educational use is the level of foreknowledge: the user may be a beginner or an advanced user. If supported by the application, this would certainly affect its behavior.

The most challenging component in the architecture is the application control block. It is responsible for finding or creating a suitable response for the request. In a simple case, it could be a list of pages that together form the CBE application accompanied with a control logic that describes the order of the pages. In other words, it defines the application with respect to the current request and request history. For example, if the application has been divided into several topics, and one topic consists of several web pages, the application control block makes sure that pages within a topic, and topics themselves, are given in the correct order.

With interactive applications, that is, when users provide parameters or data along with their requests, it is up to the application control component to validate that information. Technically, this is a very challenging question and it is discussed in more detail in the next section.

The content itself is either stored in a database or created on-the-fly. In any case, the response is always created on-the-fly, but at least parts of the content can be stored in a database. The application control block contains the necessary indexing keys for getting the content from a database or rules for creating the content. Additional information, metadata, can be used for making a search in the content, for example finding additional material or similar pages to previous ones.

Finally, the presentation control block transforms the content into suitable form with respect to user's preferences and capabilities of his/her current browser. The agent also adds session parameters etc. to the response so that for instance links point back to correct session and so on.

From functional point of view, it is a matter of opinion whether the double agent is defined as a structure containing all the described functional blocks or are the application control block and presentation block considered to be separate blocks. With implementation, separation would be useful in any case, but from the user's point of view, the whole

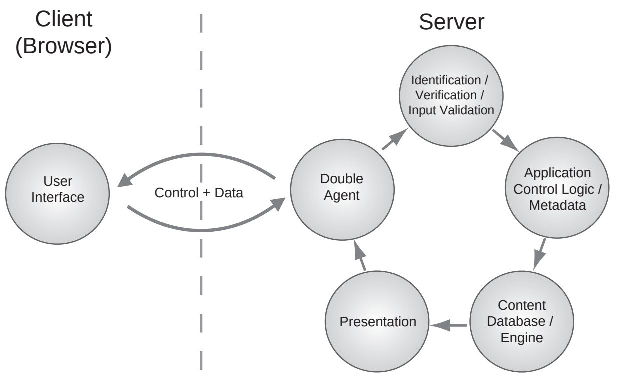

Client
(Browser)
Server
User Interface
Control + Data
Double Agent
Identification / Verification / Input Validation
Application Control Logic / Metadata
Content Database / Engine
Presentation

Figure 5.2: Request operation chain

functionality appears to be done by the agent.

Another interesting way to describe the functionality of the system is to think that it consists of two or more agents. For example dividing the functional agent structure into two separate agents, the user agent and application agent, gives another meaning to the double agent definition. Even more agents can be included: the error agent, presentation agent etc. Still, the multi-agent formalism doesn't change the basic dual nature of the system that is experienced by the user.

5.2.2 Operational Description

A typical operation chain of the architecture is presented in Figure 5.2. It starts with the agent receiving a request from the user. The request is identified, verified and passed on to the application control level. Based on the information given by the agent, appropriate content is fetched from a database or created on-the-fly. Finally, the response is created by transforming the content into appropriate presentation.

Furthermore, there are three special cases of operation: logging into the system, logging out of the system and errors. Logging into the system requires additional functionality for authenticating the user and preparing the session control. This information forms the basis for later operations. In addition, mechanisms for creating a new user account, changing passwords or closing a user account are needed as well as defining what the user is allowed to do, in other words, what is the current role of the user.

The logging out phase is another special case of interest: a mechanism is needed for terminating the session and saving the current status of the user. In a normal situation it would be expected that the user explicitly logs out from the system. However, this can't be trusted and, therefore, a timing criteria etc. is needed for determining whether the user is still using the application or not.

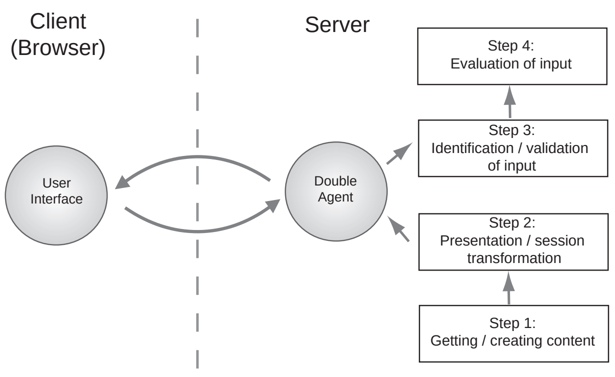

Client
(Browser)
Server
User Interface
Double Agent
Step 4:
Evaluation of input
Step 3:
Identification / validation of input
Step 2:
Presentation / session transformation
Step 1:
Getting / creating content

Figure 5.3: Process chain for interactive operation

Error situations are particularly difficult because there are two basic kinds of errors: system level errors and content-oriented errors. System level errors include errors such as session control violations, security attacks and server malfunctions. Content-oriented or application level errors are related to the content: malformed input parameters are a typical example. In a usual operation, it would be the responsibility of the application control block to provide the agent with means for determining content-oriented errors.

Many of these operations are application-specific. For instance, verifying the input or saving the status of the user when logging out, requires operations strictly dependent of the application that is used. This makes the implementation of the system quite demanding. On the other hand, session control and authentication can be implemented in a generic way and in fact, such operations are often included in existing development tools.

5.2.3 Operation with Interactive CBE-applications

Interactive operation requires some further explanation due to various implementation technologies. As discussed earlier, there are three different possibilities for creating interactivity: server-side, client-side or a combination of both server- and client-side technologies.

The most widely used solution is to use server-side technologies and limit client-side operations to input fields such as HTML forms. This approach has the advantage that the application provider has complete control over the interaction process. Another benefit is that the processing does not depend on the client software properties. In this case, the interaction is based on and limited to rather simple parameter transfer. However, in many applications this would be sufficient.

The basic server-side interactive operation involves four steps (Figure 5.3): creation of interactive content, modifying it to match current presentation and session requirements, sending the content to the user, receiving another request, validation of incoming parameters and evaluation of the actual content sent by the user. Note that process actually starts with the first request made by the user. The numbering of steps in figure 5.3 is only used to underline that the content defines the possibilities for interactivity. The process is actually circular.

An example will clarify the situation: A CBE-application includes predefined true/false or multiple-choice questions. First the questions are fetched from a database. Secondly, they are inserted into HTML template file and it is verified that the links point back to the agent. Possibly, some session control parameters are added. In the third step, the user sends the answer back. The input is then first checked that it matches the current session parameters and secondly that the parameters are indeed true/false-answers or valid choices. Finally, the input needs to be evaluated with respect to the content: are the answers correct and how to respond, if they are not.

A more challenging situation is when the question parameters are created on-the-fly. In that case, the operation chain is the same except that the questions are not fetched directly from a database but instead created or varied on-the-fly using some rule-based or similar processing.

Sometimes this kind of interaction is not satisfactory, however. The use of client-side solutions provides additional functionality that can be of interest. Typical examples are animations, simulations and graphical games. If this kind of interactivity is wanted, client-side technologies are needed. This requires even more operations from the architecture, because not all clients provide the necessary capabilities and the client-side elements need to communicate with the agent.

An example illustrates the situation: The content is a standard web page with a simulation made as a Java Applet as part of it. Now, the operation chain requires an additional step, verification that the client-software actually is capable of presenting Java Applets [15]. If it is not, the user has to be informed or alternative content created instead. Of course, the testing of browser's capabilities can and should be included in the logging in phase.

Another difficulty with the client-side technologies relates to getting and transferring input from the user. Basically, client-side technologies do not transfer any input back to the server unless specifically programmed to do so. This is not even possible in all situations. Even if they would, the communication needs to go through the agent. This requires that the client-side element can be adjusted with an interface between the element and the agent. Once again, an example illustrates the issue. A Java Applet simulation is programmed to send information about the user's operations back to the server. Because all communications go through the agent, the Applet needs to be adjusted with the agent address and session parameters. Furthermore, a specific interface needs to be created so that the agent understands that the input comes from the Applet. In the case of Java Applets and Flash applications this can be done using initialization parameters located inside the appropriate object/embed/applet-tags or use the built-in networking routines to fetch parameters separately, but other client-side technologies might not have these capabilities. If so, the only choice would be to create the whole active element on-the-fly. In practice, this might be impossible.

5.2.4 Designing Interactive Content

Thinking of the discussion above, it is obvious that even though the proposed architecture provides guidelines for design and implementation, it has to be taken into consideration already in designing interactive content.

Perhaps the most difficult task is to describe the internal logic of the CBE application in a way that the double agent can use it. In a design phase this requires detailed modeling of the interaction processes.

Another tedious design problem is to cope with various content-related error situations. For example for any question-like task there has to be a clear understanding of how to evaluate the answer and what to do if the answer is correct or not.

The use of metadata is also an important question already in the design phase. Should the material be accessible through metadata-based search and if so, what metadata would be needed to describe the content in a sufficient manner?

Despite of these problematic issues, one needs to realize that they are more of a systematizing factor rather than a hindrance. In fact, to some extent, the architecture allows software tools to be built for helping with the content design, after all, any content module needs to be described internally in the system anyway. All and all, taking these issues into account already in the design phase will result in clear and robust content modules.

5.3 Log-based Evaluation

One of the benefits of the proposed architecture is the ability to use log information for content adaptation and evaluation of learning [94], see Figure 5.4.

As a built-in feature of the architecture, it is continuously monitoring user's request history and logging that information. In fact, the session history is used for on-line content adaptation by using it partly for determining what content is provided next. This way, the user can be guided throughout the learning process. For example, if the content is divided into blocks that can be studied separately but still maintain certain hierarchy, the user can be proposed to study certain blocks before moving onto others. A more advanced example would be the use of tests before allowing the user to access more advanced material. Other criteria such as timing, can also be used.

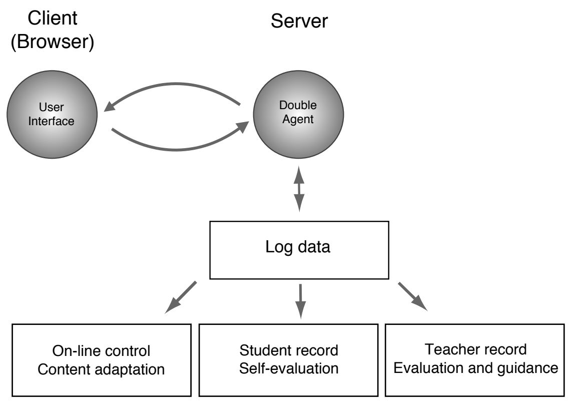


Figure 5.4: The log information can be used for three kinds of learning evaluation

The log can also be given to the user after the session. Not only it provides the user with statistical information but helps identifying content that has taken longer to go through and a picture of the whole session and navigation structure. Thus the architecture provides also a valuable tool for self-evaluation, an important part of learning as mentioned before. In fact, the user may automatically be given a report or certificate that he/she has studied the following content. Such a report could include even quite detailed information such as content descriptions, usage of time and navigation path through the material.

Furthermore, the log can be used by a teacher for exactly the same purpose. This way the teacher can get a picture of the students' learning process and be able to provide further guidance and instructions for students. By analyzing the logs, the teacher will also get details and statistical information that will help developing the material further on.

From learning point of view, the use of logs for describing and evaluating the learning process is the strongest benefit of the proposed architecture.

5.4 Implementation

There are various solutions and ways to implement self-study CBE applications. The proposed architecture is only an abstraction that emphasizes the interaction process between user and CBE application. To some extent, it makes the implementation more complex, on the other hand it provides a systematic approach for design and implementation.

Questions such as what would be the best way to implement the architecture or what tools should be used, are large and complex issues. The implementation problematics can be divided at least into the following categories:

- Modeling of educational content and content relations

- Modeling of interaction processes
- Web-based User interface
- DSP computation
- Audible and visual presentation of sound
- Presentation of mathematics
- Combined use of different technologies
- Security considerations

Discussion of these topics can be found next in chapter 6.

Chapter 6

A Case Discussion

This chapter discusses the applicability and implementation of the architecture in general and in a particular case of a tutorial level CBE application regarding digital signal processing concepts. The goal is to point out and emphasize the application specific requirements and implementation problematics through a case study.

6.1 Case: Introduction to Signal Processing

"Introduction to Signal Processing" was a tutorial level CBE application developed by the author as a Master's Thesis project in 1995-1996 [92]. The CBE application was designed for 1st to 3rd year university-level engineering students. A background with mathematics and physics was assumed but the students were not expected to have any prior knowledge of signal processing. The application was used in a course "Fundamentals of Acoustics I" at the Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing. The application was additional material and was not an obligatory part of the course requirements. During two years time, the application was used by 40-50 students.

The "Introduction to Signal Processing" was a stand-alone application running on top of Macintosh Common Lisp software and QuickSig DSP environment [92]. It was running on a single Apple Macintosh computer located in the laboratory. No network connections or resources were used. The graphical user interface was also completely Lisp-based, even though it included basic hypermedia functionality such as hyperlinks within the application.

The starting point for this project was to investigate if the application could be used and distributed through the web instead of a stand-alone workstation. However, it turned out to be more difficult than expected. As discussed in this chapter, even today we are still facing some of the same technical questions that we had when the project started.

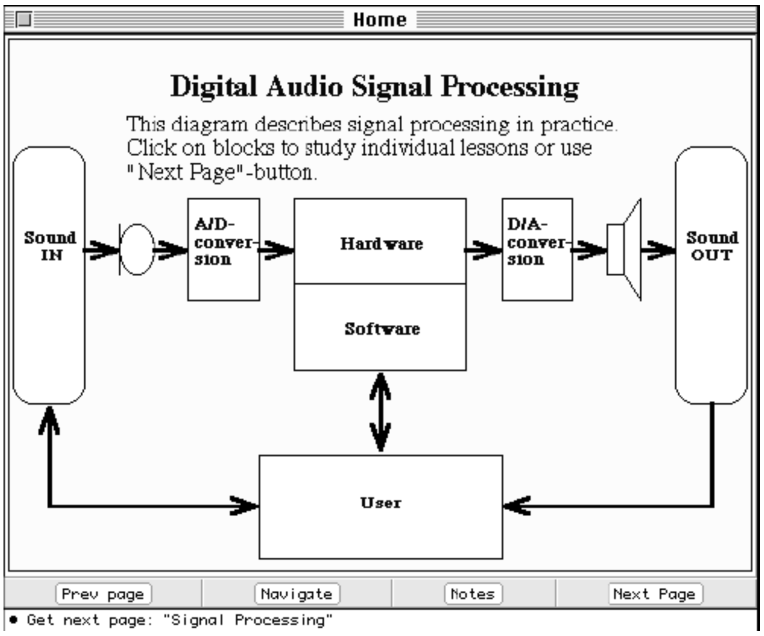

Home
Digital Audio Signal Processing
This diagram describes signal processing in practice. Click on blocks to study individual lessons or use "Next Page"-button.
Sound IN
A/D-conver-sion
Hardware
Software
D/A-conver-sion
Sound OUT
User
Prev page
Navigate
Notes
Next Page
• Get next page: "Signal Processing"

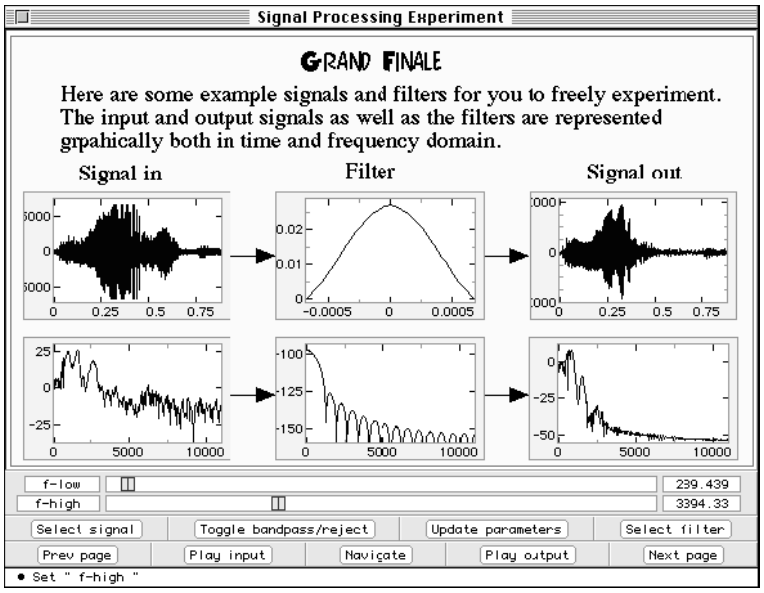
Figure 6.1: Main view of the "Introduction to Signal Processing" CBE application

6.1.1 Application Description

The "Introduction to Signal Processing" CBE application covered some fundamental topics in digital signal processing such as signal properties and presentations, analog-to-digital (A/D)- and digital-to-analog (D/A)-conversion and digital filters. The user interface of the application was a block diagram presented in Figure 6.1. Each block represented a topic (e.g "chapter") that the student could choose by clicking on the block. After going through a content block, the student returned to the same view allowing he/she to continue with another topic. Furthermore, a default path was made available so that the students could also navigate using only back- and forth-buttons.

Each content block consisted of several pages of text, figures and audio samples along with simple questions and interactive tasks such as filtering examples etc. After studying through all the topics, the student was allowed to enter the "Grand Finale", an interactive experiment summarizing most of the application content (see Figure 6.2). This experiment allowed the student to perform digital filtering with different basic filter types, adjust the filter parameters, see both time and frequency domain representations of the signals and listen to the signals before and after the filtering operation.

The "Introduction to Signal Processing" CBE application was designed using the design method for computer-based learning environment by Lifländer [66], which was based on pedagogical theory by Engeström [38]. This theory is focused only on the cognitive learning


Signal Processing Experiment

GRAND FINALE

Here are some example signals and filters for you to freely experiment. The input and output signals as well as the filters are represented grpahically both in time and frequency domain.

f-low 239.439

f-high 3394.33

● Set " f-high "

Figure 6.2: Final view of the "Introduction to Signal Processing" CBE application

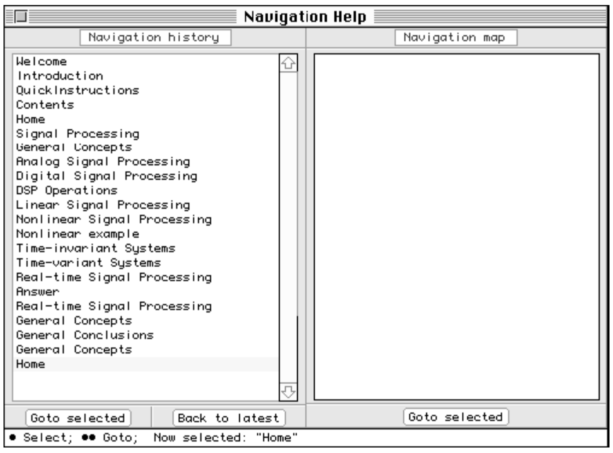
process and does not take social aspects of learning into account. Nowadays, the theory has been superseded with the constructivistic view of learning, but at the time, it was a valid and useful model especially for self-study CBE applications.

6.2 Discussion of Implementation Issues

There are several possibilities and technologies to implement the double-agent architecture. Based on my personal experiences, I would choose some commonly used web programming language and server-based solution as the basis of implementation. One of the most important requirements is a large availability of programming libraries or modules, which make Perl, PHP and Java candidates of special interest. For information storage, databases are needed. SQL or other relational databases would be of interest, because they are widely supported in web application development. In addition, XML would probably be needed for the representation of internal data structures.

6.2.1 Content and Interaction Modeling

With the implementation, perhaps the most difficult question is how an educational application and its internal structure should be described in terms of software. In other words, how to model and describe the interaction processes and related control mechanisms. After all, the internal structure and the interaction processes are very content-oriented



```
Welcome
Introduction
QuickInstructions
Contents
Home
Signal Processing
General Concepts
Analog Signal Processing
Digital Signal Processing
DSP Operations
Linear Signal Processing
Nonlinear Signal Processing
Nonlinear example
Time-invariant Systems
Time-variant Systems
Real-time Signal Processing
Answer
Real-time Signal Processing
General Concepts
General Conclusions
General Concepts
Home
```

```
● Select; ●● Goto;  Now selected: "Home"
```

Figure 6.3: Navigation history for the "Introduction to Signal Processing" CBE application

and application-dependent. In a simple case such as an interactive demo, the internal structure is fairly easy but the more content there is, the more complex will the internal relations get. Also coping with error situations is a true challenge for complex interactive material.

At the moment, there seems to be no appropriate specifications for describing advanced content structures and interaction that would be directly applicable here. The Unified Modeling Language (UML) [4] is a commonly used tool for describing interaction processes and even business models. However, UML was designed specifically as a design language, not implementation, which would be needed here.

AHA! [3] presents a model and tools for creating concepts and rules. Whether this would be suitable for more advanced interaction processes, would probably be worth investigating.

In comparison, the internal structure of the "Introduction to Signal Processing" CBE application was defined by Lisp objects and functions. When the last page of each content block was reached, it was checked if all pages of that block had been visited and the whole block could be marked as visited. Every time the student returned to the main view, a function was used for determining which blocks were left to study or if they all had been already visited and the student allowed to continue to the "Grand Finale".

6.2.2 Navigation and On-line Adaptation

The "Introduction to Signal Processing" CBE application continuously kept a record of visited pages in memory. The navigation history was also available to the user (see Figure 6.3). The user could return to any visited page by choosing the page from a list. This functionality is similar to the "Back" and "Forward"-navigation of web browsers. In addition, the application included a navigation map similar to "sitemaps" available on many websites.

The internal structure of the application was so simple that there was no real challenge in terms of on-line adaptation: the simple "chapter"-checks, the default path, which was a list of pages in predefined order and the navigation history, which kept record of the visited pages were more than enough to cover the users' needs to navigate within the application. Furthermore, all pages had been designed so that they were unique in terms of content, in other words, each page could be accessed only within one content block and there were no conflicting paths to pages. In addition, the number of pages and topics was so small that the whole application structure and content relations could be described as a simple list of pages, chapters and visited-flags and kept in memory the whole time the application was used.

Similar functionality could be achieved with a web-based implementation as well. However, because of the request-based nature of the web, one needs to use session control techniques to ensure that each request can be mapped correctly with the internal structure of the application and navigation works expectedly. Because, by default, multiple users could access the application simultaneously, user identification needs to be combined with the session control. Furthermore, in a multi-user environment, one needs to control and limit memory resources with respect to the amount of users. Therefore, even though a simple content structure and navigation rules could be kept in active memory using session control and persistent server-side techniques, one also needs to identify errors and orphan sessions and clean up the memory respectively. In a general case, one needs to write the whole sessions and application state into a database or file and read it back in when needed. In fact, this is one of the core functionalities of the double-agent.

6.2.3 User Interfaces

Even though the structure for pages in the "Introduction to Signal Processing" CBE application was quite simple, it's user interface included some advanced signal processing oriented functionality. For instance one could make selections, zoom and scale every signal view on a page. In case of time domain signal representations, one could also listen to audio signals or selections of signals. This type of functionality is impossible to implement using only standard HTML and web browser capabilities. As a matter of fact, this kind of intelligent views require vector-based graphics whereas the standard web graphics are just bitmaps.

Therefore, the question of implementing this kind of user interfaces is dependent of the client-side technologies and features of web browsers. Flash [2] or Java Applets [106] would provide necessary capabilities for manipulating the views and possibly playing audio. However, even though both of these plugin-based technologies are widely used and freely available, the user needs to have them installed and might experience strange problems due to his/her own browser setup [15].

Also the W3C Scalable Vector Graphics (SVG) [127] technology would be of special interest as a tool for user interface graphics such as signal presentations etc. SVG not only provides a vector-based presentation of graphics but also allows Javascript to be used for manipulating the graphical elements. Unfortunately, the browser support to SVG and related Javascript functionality varies a lot.

Recently, the Ajax [46] technology has gained a lot of popularity in a short time. The technology allows parts of a web page to be dynamically updated without the need for creating the whole page again, which is a typical case with server-side technologies. Ajax utilizes a number of technologies for doing this, especially Javascript [49] and XML-HttpRequest [73].

The W3C has some interesting related work going on at the moment: the Web APIs Working Group [128] and the Web Application Formats Working Group [129] are working on specifications and languages for client-side web application development. At this point, however, it is too early to say how this work will affect server-side driven technology such as the proposed architecture.

6.2.4 Acoustics and Digital Signal Processing

One should not forget the technical challenges related to audio and signal processing content. Meaningful educational content will require actual digital signal processing performed within a reasonable response time and delivery of audible content. Because the "Introduction to Signal Processing" CBE application was built on top of the QuickSig DSP environment, any audio signal processing task could be mapped directly with the user interface. In terms of response time, the only limitation was the computational power of the Apple Macintosh workstation. For that particular application, all the DSP calculations could be done with reasonable delay (reasonable meaning not real-time but not disturbing either). Therefore, interactive DSP experiments such as "Grand Finale" (Figure 6.2) were easy and straightforward to implement.

On the web, implementing a similar experiment is far from being easy and straightforward. One could use only client-side technologies such as Flash or Java Applets, but this approach would lack the integration and contextual relations to larger CBE material, not to mention the technical problems of client-side technologies as mentioned before.

The server-side technologies have their own challenges and problems as well. First of all, one needs to have access to computational resources. There are, for example, DSP libraries available for several programming languages that could be accessed through server-side programming. But in order to do this, one needs to map the user interface operations into application tasks and further on into appropriate subroutine calls. Furthermore, one needs to take care of possible temporary files needed for the DSP operations etc. On the other hand, this way the DSP operations can be tightly integrated into the application and the agent.

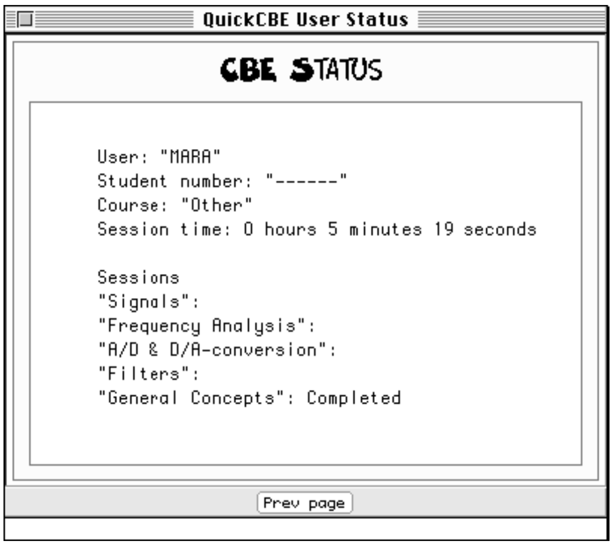
From the beginning of this project, one particular question has been problematic: the use of audio input. For audio signal processing education, the use of the student's own audio samples such as speech, would definitely be beneficial in terms of understanding what is going on. Currently, there is no web browser that would support recording of audio directly. This is not only a technical question, but it also has important security and privacy concerns. However, some plugin-operated technologies such as Java Applets [106], especially JSyn [100] and Flash [2] would be able to record sound through the web browser, if the user allows it by adjusting some security settings and preferences. Otherwise, the only possibility would be to guide the user to use separate software for recording sound or preparing their own sound samples and use the file upload mechanism of the web browser in the web application.

6.2.5 Mathematics

Another DSP-related problem is the presentation and use of mathematics with a web browser. Mathematics are a fundamental part of university-level engineering education and, therefore, the use of mathematics has an important role also in CBE. The "Introduction to Signal Processing" CBE application included some mathematical formulas and calculation tasks, but only on an introductionary level. The formulas were presented as figures and used special fonts. No interactive operations were linked to formulas or mathematical expressions.

For the web implementation, the first problem is how to present mathematical expressions on a web page. For publications etc. the most common way has been to use bitmap graphics. Unfortunately, this approach lacks all interaction capabilities making it suitable only for viewing formulas. What is needed, is a browser-independent way of defining and presenting mathematics. The Mathematical Markup Language (MathML) [125], a W3C recommendation, tries to do exactly this. The XML-based MathML is also natively supported by many of the recent browsers making it a very interesting candidate for CBE purposes as well. Furthermore, the specification is also supported by many manufacturers of mathematical software.

The second step would be to edit and evaluate mathematics within the CBE application. Once again, simple demonstrations etc. could be implemented with client-side only technologies. However, a more flexible solution would include the use of mathematical computations on the server-side. Similarly to the DSP computations, this could be done with the help of various mathematical programming libraries. One needs to keep in mind though that in DSP, it is often needed to provide graphical presentations or audio signals as responses to input given as mathematical expressions. Therefore, additional DSP, visualization and



```
User: "MARA"
Student number: "------"
Course: "Other"
Session time: 0 hours 5 minutes 19 seconds

Sessions
"Signals":
"Frequency Analysis":
"A/D & D/A-conversion":
"Filters":
"General Concepts": Completed
```

Figure 6.4: Status screen for the "Introduction to Signal Processing" CBE application

audio libraries need to be used together.

Another interesting approach would be to use an external server for mathematical computations. At least Matlab® [69] and Mathematica® [133] have also developed server-oriented computational tools and APIs for using them over the network. However, integrating that technology into the architecture or CBE applications might be a tedious task.

Discussion of the use of mathematics on the web can be found in an article by Foster [42].

6.2.6 Logs and Evaluation of Learning

The "Introduction to Signal Processing" CBE application included a simple logging mechanism that was mostly created for debugging purposes. However, it turned out to be a valuable source of information when the use of the application was later analyzed. The log system recorded the time when a page was loaded together with the page title. With such a simple log, it was possible to calculate the use of time for the whole application as well as for each page. Furthermore, the log showed the route that the student had used in order to reach the "Grand Finale", i.e. the order of chapters.

The log information was not available to the student directly, but instead, the student was able to monitor how much time they had spent using the application and which chapters they had already studied and which not (see Figure 6.4).

On the web, logs can be created and accessed with any server-side technology. In fact, most web servers are configured to log requests and errors for system administration purposes. However, these standard logs are not directly suitable for CBE evaluation purposes. Instead, more specific logs are needed.

First of all, the requests need to be mapped against the context in order to get a picture of the navigation path. In practice, unique identifiers are needed already for accessing the pages with the agent, which makes logging them quite straightforward. Secondly, the session information needs to be logged together with the content identifiers in order to combine the requests and users and, finally, a timestamp is needed for keeping the log in the correct order and calculating the time usage.

However, one needs to understand that calculating the use of time is not completely reliable due to the request-based nature of the web. There is no way of knowing for sure if the user visits some other pages during the session, even though some clever client-side techniques can be used for identifying if the user leaves a page. Typically, modern web browsers allow users to use multiple windows or tabs for surfing making it impossible to know if they have visited other pages. Therefore, the use of time should be carefully considered if used as an evaluation criteria.

An interesting trade-off is how much and what information is logged? As discussed earlier, small and compact navigation history is needed for on-line adaptation. On the other hand, for off-line evaluation, the more information is logged, the more advanced analysis and evaluation can be performed. From a practical point of view, the logs can be kept fairly simple and compact with proper session control mechanism.

When we think about evaluation of learning, the key is to be able to map the log information to the educational content. Obviously, this is very application dependent and requires knowledge of the content, but providing the information for doing this mapping is exactly what the double-agent architecture does!

Chapter 7

Conclusions

This thesis has discussed web-based interactive self-study applications for acoustics and audio signal processing. Background for learning and education in general, special requirements of acoustics and signal processing education and finally web technology has been presented. Furthermore, evaluation of learning was discussed.

Based on this background, a system level architecture for web-based interactive applications was proposed and explained. The double-agent architecture, as it is called, uses a software agent to act as an interface between the user and the content. Not only does the agent represent the user by taking care of each request, it uses various methods for evaluating requests and providing content thus representing a teacher as well as the user.

The benefits of this architecture include a controlled interaction process, necessary for self-study educational applications and a built-in log feature which is utilized for intelligent content adaptation. Furthermore, these logs may be used by both learners and teachers for evaluation of learning.

The on-going and partly rapid development of web technologies has affected the project from the very beginning. One of the key factors is the development of the web browsers. From user's point of view, the browser is the key tool for accessing resources and using the applications. The browser is also a personal tool with many personal settings and preferences. Therefore, the use of additional software in a web application (for instance plugins), should be considered and evaluated carefully beforehand. In some cases it might not even be possible to use plugins etc. for technical reasons. After all, nowadays there is a wide variety of platforms and operating systems that are used for running web browsers, game consoles and mobile phones, just to mention a couple of examples. However, one needs to realize that there is no "standard" web browser that could be used as a reference. All web browsers are different, both in terms of versions as well as platforms. Some of the browsers are simply more popular than others.

It is clear that for implementing advanced interactive applications and web user interfaces, not one but many technologies need to be used together. In fact, in my opinion the most difficult technical problems with implementation would be the cooperative use of different client-side technologies and how to use them in a controlled fashion together with the server side architecture.

All and all, the double-agent architecture provides interesting possibilities in terms of evaluation of learning and content adaptation but further research is still needed for solving many practical implementation questions.

Bibliography

[1] Adobe Systems Inc. Adobe PDF Technology Center, 2005. http://partners.adobe.com/public/developer/pdf/topic.html.

[2] Adobe Systems Inc. Flash Developer Center, 2005. http://www.macromedia.com/devnet/flash/.

[3] AHA Project. AHA! project: Adaptive Hypermedia for All, 2006. http://aha.win.tue.nl/.

[4] Sinan Si Alhir. Understanding the Unified Modeling Language. *Methods & Tools*, 7(1):11–18, Spring 1999. http://home.comcast.net/~salhir/UnderstandingTheUML.PDF.

[5] Marcos Alonso, Günter Geiger, and Sergi Jordá. An Internet Browser Plug-in for Real-time Sound Synthesis using Pure Data. In *Proceedings of the International Computer Music Conference (ICMC 2004)*, Miami, Florida, USA, November 1-6 2004.

[6] Anon. Music-DSP Source Code Archive. http://www.musicdsp.org/.

[7] Anon. The CMS Matrix, 2005. http://www.cmsmatrix.org/.

[8] Anon. Wiki software, 2005. http://en.wikipedia.org/wiki/Wiki_software.

[9] Anon. Wikipedia, the Free Encyclopedia, 2006. http://www.wikipedia.org.

[10] Apache Software Foundation. Apache HTTP Server Project, 1999-2005. http://httpd.apache.org/.

[11] Apache Software Foundation. Mod_perl, 2005. http://perl.apache.org/.

[12] Apache Software Foundation. Mod_unique_id, 2005. http://httpd.apache.org/docs/1.3/mod/mod_unique_id.html.

[13] Apple Computer Inc. Documentation: QuickTime Internet & Web, 2005. http://developer.apple.com/documentation/QuickTime/InternetWeb-date.html.

[14] Ariadne Foundation. ARIADNE Foundation for the European Knowledge Pool, 2004. http://www.ariadne-eu.org/.

[15] Eduardo Garcia Barrachina. Interactive On-line Testing of Java and Audio Support on Web Browsers. Master's thesis, Helsinki University of Technology, Faculty of Electrical Engineering, Laboratory of Acoustics and Audio Signal Processing, 2003.

[16] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax, January 2005. ftp://ftp.rfc-editor.org/in-notes/rfc3986.txt.

[17] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web, A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May 2001. http://www.scientificamerican.com/linktous.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[18] Bert Bos. Cascading Style Sheets Home Page, 2005. http://www.w3.org/Style/CSS/.

[19] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions, June 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3546.txt.

[20] Benjamin S. Bloom and David R. Krathwohl. *Taxonomy of Educational Objectives: The Classification of Educational Goals, by a committee of college and university examiners. Handbook I: Cognitive Domain*. Longmans, Green, New York, USA, 1956.

[21] Monica Bonett. Personalization of Web Services: Opportunities and Challenges. *Ariadne*, (28), June 2001. http://www.ariadne.ac.uk/issue28/personalization/.

[22] Paul De Bra, Natalia Stash, and David Smits. Creating Adaptive Applications with AHA!, Tutorial for AHA! version 3.0. In *International Conference on Adaptive Hy-*

permedia (AH2004), Eindhoven, Netherlands, August 2004. http://www.win.tue.nl/~debra/ah2004/tutorial.pdf.

[23] Jerome Bruner. *The Process of Education*. Harvard University Press, Cambridge, MA, USA, 1960.

[24] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996. http://www2.sis.pitt.edu/~peterb/papers/UMUAI96.pdf.

[25] Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User Adapted Interaction, Ten Year Anniversary Issue (Alfred Kobsa, ed.)*, 11 (1/2):87–110, 2001. http://www2.sis.pitt.edu/~peterb/papers/brusilovsky-umuai-2001.pdf.

[26] Vannevar Bush. As We May Think. *The Atlantic Monthly*, July 1945. http://www.theatlantic.com/doc/194507/bush.

[27] CERT. CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests, February 2000. http://www.cert.org/advisories/CA-2000-02.html.

[28] P. Coburn, P. Kelman, N. Roberts, T.F.F. Snyder, D.H. Watt, and C. Weiner. *Practical guide to computers in education*. Addison-Wesley, USA, 1982.

[29] Cycling '74 Inc. Max/MSP, 2005. http://www.cycling74.com/products/maxmsp.html.

[30] DARPA. The DARPA Agent Markup Language Homepage, 2005. http://www.daml.org.

[31] Martin Dougiamas. A journey into Constructivism, November 1998. http://dougiamas.com/writing/constructivism.html.

[32] Martin Dougiamas. Moodle, 2005. http://moodle.org/mod/resource/view.php?id=3849.

[33] Dublin Core Metadata Initiative. Dublin Core Metadata, 2005. http://dublincore.org/.

[34] François Déchelle, Riccardo Borghesi, Maurizio de Cecco, Enzo Maggi, Joseph B. Rovan, and Norbert Schnell. jMax: a new JAVA-based editing and control system

for real-time musical applications. In *ICMC: International Computer Music Conference*, Ann Arbor, USA, Octobre 1998. http://freesoftware.ircam.fr/rubrique.php3?id_rubrique=14.

[35] John W. Eaton. GNU Octave, 1998. http://www.octave.org.

[36] ECMA. Standard ECMA-262: ECMAScript Language Specification, 3rd edition, December 1999. http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[37] education.au limited. The Global Learning Objects Brokered Exchange (GLOBE), 2004. http://globe.edna.edu.au/globe/go/cache/offonce/pid/2.

[38] Yrjö Engeström. *Learning by Expanding: An Activity - Theoretical Approach to Developmental Research*. Helsinki, 1987. http://lchc.ucsd.edu/MCA/Paper/Engestrom/expanding/toc.htm.

[39] Jacques Ferber. *Multi-Agent Systems*. Addison-Wesley, 1 edition, 1999.

[40] R. Fielding, J. Gettys, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, June 1999. ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt.

[41] Bjarne A. Foss, Kjell E. Malvig, and Tor I. Eikaas. Remote Experimentation - New Content in Distance Learning. In *International Conference on Engineering Education (ICEE2001)*, Oslo, Norway, August 6-10 2001. IEEE.

[42] Kenneth R. Foster. Math on the Internet. *IEEE Spectrum*, 36(4):36–40, April 1999.

[43] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication, June 1999. ftp://ftp.rfc-editor.org/in-notes/rfc2617.txt.

[44] Christopher Frauenberger and Winfried Ritsch. A Real-time Audio Rendering System for the Internet (iARS), Embedded in an Electronic Music Library (IAEM). In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, London, UK, September 8-11 2003. http://iem.iaem.at/doku/iARS/.

[45] Stephen Gance. Are constructivism and computer-based learning environments incompatible? *Journal of the Association for History and Computing*, V(1), May 2002. http://mcel.pacificu.edu/JAHC/JAHCV1/K-12/gance.html.

[46] Jesse James Garrett. Ajax: A New Approach to Web Applications, February 2005. http://www.adaptivepath.com/publications/essays/archives/000385.php.

[47] Dean Gaudet. Cookies, Dialogues and Tracking, 1996. http://www.arctic.org/~dean/cookies.html.

[48] George S. Georgiev, Hubert Roth, Silvia Stefanova, Georgi T. Georgiev, Emil Stoyanov, and Otto Rösch. How and why to build and use virtual laboratories. *World Transactions on Engineering and Technology Education*, 1(2):191–196, 2002.

[49] Danny Goodman. *JavaScript Bible 4th ed.* Hungry Minds, USA, 2001.

[50] Marty Hall and Larry Brown. *Core Servlets and JavaServer Pages, Vol. 1: Core Technologies*. Sun Microsystems Press/Prentice Hall, USA, 2 edition, 2004. http://volume1.coreservlets.com/.

[51] Helsinki University Centre for Research on Networked Learning and Knowledge Building. Development of Learning Theories, 2004. http://www.helsinki.fi/science/networkedlearning/eng/delete.html.

[52] Hyperwave AG. HyperWave, 2005. http://www.hyperwave.com/e/solutions/horizontal/elearning/.

[53] IBM Corporation. Web Services Reliable Messaging, 2005. http://www.ibm.com/developerworks/library/specification/ws-rm/.

[54] IEEE Learning Technology Standards Committee. IEEE Standard for Learning Object Metadata, 2002. http://grouper.ieee.org/groups/ltsc/wg12/index.html.

[55] Emmanuel C. Ifeachor and Barrie W. Jervis. *Digital Signal Processing: A Practical Approach*. Pearson, USA, 2 edition, 2001.

[56] IMS Global Learning Consortium Inc. IMS Learning Resource Meta-data Specification, 2005. http://www.imsglobal.org/metadata/index.html.

[57] Leander Kahney. HyperCard Forgotten, but Not Gone. *Wired News*, 2002. http://www.wired.com/news/mac/0,2125,54365,00.html.

[58] Matti Karjalainen. DSP Software Integration by Object-Oriented Programming: A Case Study of QuickSig. *IEEE ASSP Magazine*, pages 21–31, April 1990.

[59] Matti Karjalainen. BlockCompiler - A research Tool for Physical Modeling and DSP. In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, pages 264–269, London, UK, September 8-11 2003.

[60] Matti Karjalainen and Martti Rahkila. Learning signal processing concepts and psychoacoustics in the QuickSig DSP environment. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP95)*, pages 1125–1128, Detroit, Michigan, USA, May 9-12 1995. IEEE.

[61] Harri Ketamo. *User and Platform Adaptation in Web-based Learning Environments*. PhD thesis, Tampere University of Technology, 2002. Tampere University of Technology Publications 381.

[62] Kevin Kline, Daniel Kline, and Brand Hunt. *SQL in a Nutshell*. O'Reilly, USA, 2 edition, 2004. http://www.oreilly.com/catalog/sqlnut2/.

[63] Marja Kopponen. *CAI in CS*. PhD thesis, University of Joensuu, 1997. Computer Science Dissertations 1.

[64] D. Kristol and L. Montulli. HTTP State Management Mechanism, October 2000. ftp://ftp.rfc-editor.org/in-notes/rfc2965.txt.

[65] Rasmus Lerdorf and Kevin Tatroe. *Programming PHP*. O'Reilly, USA, 1 edition, 2002. http://www.oreilly.com/catalog/progphp/.

[66] Veli-Pekka Lifländer. A designing method for a computer-based learning environment. In *ADCIS 30th Conference*, Philadelphia, USA, 1988.

[67] Miroslav D. Lutovac, Dejan V. Tosic, and Brian L. Evans. *Filter Design for Signal Processing using MATLAB®and Mathematica®*. Prentice-Hall, USA, 2000.

[68] Massachusetts Institute of Technology. MIT OpenCourseWare, 2005. http://ocw.mit.edu/OcwWeb/Global/AboutOCW/about-ocw.htm.

[69] Mathworks Inc. Matlab, 1994-2005. http://www.mathworks.com/products/matlab/.

[70] Mathworks Inc. Mathtools.net - Link Exchange for the Technical Computing Community, 2001-2005. http://www.mathtools.net/.

[71] M.R. Matthews. Constructivism in Science and Mathematics Education. In Denis C. Phillips, editor, *National Society for the Study of Education, 99th Yearbook*, pages 161–192. University of Chicago Press, Chigago, USA, 2000. http://wwwcsi.unian.it/educa/inglese/matthews.html.

[72] James H. McClellan, Ronald W. Schafer, and Mark A. Yoder. *DSP First - A Multimedia Approach*. Prentice-Hall, USA, 1998.

[73] Drew McLellan. Very Dynamic Web Interfaces, 2005. http://www.xml.com/pub/a/2005/02/09/xml-http-request.html.

[74] MDC Project. Gecko Plugin API Reference, 2005. http://developer.mozilla.org/en/docs/Gecko_Plugin_API_Reference.

[75] MDC Project. Mozilla Developer Center: Extensions, 2005. http://developer.mozilla.org/en/docs/Extensions.

[76] MDC Project. Mozilla Developer Center: Plugins, 2005. http://developer.mozilla.org/en/docs/Plugins.

[77] MDC Project. Mozilla Developer Center: XUL, 2005. http://developer.mozilla.org/en/docs/XUL.

[78] Barbara Means, John Blando, Kerry Olson, Teresa Middleton, Catherine Cobb Morocco, Arlene R. Remz, Judith Zorfass, Richard W. Riley, Sharon P. Robinson, and Joseph C. Conaty. Using technology to Support Education Reform, 1993. http://www.ed.gov/pubs/EdReformStudies/TechReforms/index.html.

[79] Microsoft Corporation. Active Server Pages, 2005. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/activeservpages.asp.

[80] Microsoft Corporation. Microsoft .NET, 2005. http://www.microsoft.com/net/default.mspx.

[81] National Instruments Inc. LabView, 2005. http://www.ni.com/labview/.

[82] NCSA HTTPd Development Team. The Common Gateway Interface, 1996. http://hoohoo.ncsa.uiuc.edu/cgi/.

[83] E. Nebel and L. Masinter. Form-based File Upload in HTML, November 1995. ftp://ftp.rfc-editor.org/in-notes/rfc1867.txt.

[84] Netscape Inc. Netscape Navigator 2.0 for Windows Release Notes, 1996. http://wp.netscape.com/eng/mozilla/2.0/relnotes/windows-2.0.html.

[85] Jakob Nielsen. *Designing Web Usability*. New Riders Publishing, USA, 1999.

[86] Jakob Nielsen. Usability 101: Introduction to Usability, 2003. http://www.useit.com/alertbox/20030825.html.

[87] OmniPilot Software, Inc. Lasso Professional Server, 2005. http://www.omnipilot.com/Lasso.1541.lasso.

[88] Sophocles J. Orfanidis. *Introduction to Signal Processing*. Prentice-Hall, USA, 1995.

[89] Miller Puckette. Pure Data: another integrated computer music environment. In *Proceedings of the Second Intercollege Computer Music Concerts*, pages 37–41, Tachikawa, Japan, 1996. http://puredata.info/.

[90] Ville Pulkki. Virtual Sound Source Positioning Using Vector Base Amplitude Panning. *Journal of the Audio Engineering Society*, 6(45), June 1997.

[91] Marti Rahkila and Jyri Huopaniemi. Real-time Internet Audio - Problems and Solutions. In *Audio Engineering Society 102nd International Convention*, Munich, Germany, March 22-25 1997. IEEE. Preprint 4477. 32 p.

[92] Martti Rahkila. A Computer Based Education System for Signal Processing. Master's thesis, Helsinki University of Technology, Faculty of Electrical Engineering, Laboratory of Acoustics and Audio Signal Processing, 1996.

[93] Martti Rahkila. Considerations of Computer Based Education of Acoustics and Signal Processing. In *Fronties in Education (FIE98)*, pages 679–684, Tempe, Arizona, USA, November 12-15 1998. IEEE.

[94] Martti Rahkila. Evaluation of Computer Based Education Using Logsystems. In *Fronties in Education (FIE99)*, San Jose, Puerto Rico, November 10-13 1999. IEEE.

[95] Martti Rahkila. A Double Agent Architecture for eLearning Applications. In *International Conference on Engineering Education (ICEE2001)*, Oslo, Norway, August 6-10 2001. IEEE.

[96] Martti Rahkila and Matti Karjalainen. An Interactive DSP Tutorial on the Web. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP97)*, pages 2253–2256, Munich, Germany, April 20-26 1997. IEEE.

[97] Martti Rahkila and Matti Karjalainen. An Experimental Architecture for Interactive Web-based DSP Education. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP98)*, pages 1857–1860, Seattle, Washington, USA, May 12-15 1998. IEEE.

[98] Martti Rahkila and Ville Pulkki. WWWVbap, 2000. http://www.acoustics.hut.fi/demos/wwwvbap/.

[99] Scripting News Inc. XML-RPC Homepage, 2005. http://www.xmlrpc.com/.

[100] Softsynth Inc. JSyn, 1997-2005. http://www.softsynth.com/jsyn/.

[101] Guy L. Steele. *Common Lisp the Language*. Digital Press, USA, 2 edition, 1990. http://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html.

[102] Lincoln D. Stein. *Web Security: A Step-by-Step Reference Guide*. Addison-Wesley, USA, 1997.

[103] Lincoln D. Stein. W3C Security Resources, 1999. http://www.w3.org/Security/.

[104] SumTotal Systems Inc. Toolbook, August 2005. http://www.toolbook.com.

[105] Sun Microsystems Inc. Java 2 Platform, Enterprise Edition (J2EE), 1994-2005. http://java.sun.com/j2ee/.

[106] Sun Microsystems Inc. Java Applets, 1994-2005. http://java.sun.com/applets/.

[107] Sun Microsystems Inc. Java Plug-in Technology, 1994-2005. http://java.sun.com/products/plugin/.

[108] Sun Microsystems Inc. Java Runtime Environment (JRE), 1994-2005. http://java.sun.com/j2se/desktopjava/jre/index.jsp.

[109] Sun Microsystems Inc. JavaSound API Programmer's Guide, 2001. http://java.sun.com/j2se/1.5.0/docs/guide/sound/programmer_guide/.

[110] Sun Microsystems Inc. Java Web Server, 2005. http://www.sun.com/software/products/web_srvr/home_web_srvr.xml.

[111] Jonathan Swartz, Dave Rolsky, Ken Williams, and John Williams. Mason, 1998-2005. http://www.masonhq.com/.

[112] The Open Source Collective Inc. Open Source CMS, 2005. http://www.opensourcecms.com/.

[113] The PHP Group. PHP, 2001-2005. http://www.php.net/.

[114] Stephen Todd, Francis Parr, and Michael Conner. A Primer for HTTPR, 2005. http://www.ibm.com/developerworks/webservices/library/ws-phtt/.

[115] Ila Tokola, Matti Karjalainen, and Martti Rahkila. A software "teacher" for acoustical measurements. In *International Conference on Acoustics (ICA98)*, pages 2077–2078, Seattle, Washington, USA, June 20-26 1998. ASA.

[116] Sam Tregar. HTML::Template - Perl module to use HTML Templates from CGI scripts, 2000-2002. http://search.cpan.org/~samtregar/HTML-Template-2.7/Template.pm.

[117] UIAH Media Lab. Fle3 Future Learning Environment, 2005. http://fle3.uiah.fi/.

[118] W3C. Web Accessibility Initiative (WAI), 1994-2005. http://www.w3.org/WAI/.

[119] W3C. Extensible Markup Language (XML), 1996-2003. http://www.w3.org/XML/.

[120] W3C. Document Object Model, 1997-2005. http://www.w3.org/DOM/.

[121] W3C. HTML 4.01 Specification, December 1999. http://www.w3.org/TR/html4/.

[122] W3C. Semantic Web Activity, 2001. http://www.w3.org/2001/sw/.

[123] W3C. XHTML? 1.0 The Extensible HyperText Markup Language (Second Edition), August 2002. http://www.w3.org/TR/xhtml1/.

[124] W3C. Web Services Description Working Group, 2002-2005. http://www.w3.org/2002/ws/desc/.

[125] W3C. Mathematical Markup language (MathML) 2.0 Specification, October 2003. http://www.w3.org/TR/MathML2/.

[126] W3C. Resource Description Framework (RDF), 2005. http://www.w3.org/RDF/.

[127] W3C. Scalable Vector Graphics (SVG), 2005. http://www.w3.org/Graphics/SVG/.

[128] W3C. Web APIs Working Group, 2005. http://www.w3.org/2006/webapi/.

[129] W3C. Web Application Formats Working Group, 2005. http://www.w3.org/2006/appformats/.

[130] W3C. XML Protocol Working Group, 2005. http://www.w3.org/2000/xp/Group/.

[131] Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl*. O'Reilly, USA, 3 edition, 2000. http://www.oreilly.com/catalog/pperl3/.

[132] WebCT Inc. WebCT, 2005. http://www.webct.com.

[133] Wolfram Research Inc. Mathematica, 2005. http://www.wolfram.com/products/mathematica/.