INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen

Oettingenstraße 67 D–80538 München

Ludwig
Maximilians–
Universität
München

LMU

Contextual Web Services for Teaching

Georg Schneemayer

Diplomarbeit

Beginn der Arbeit:	16. Mai 2002
Abgabe der Arbeit:	10. Oktober 2002
Betreuer:	Prof. Dr. François Bry
	Dr. Norbert Eisinger

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

München, den 10. Oktober 2002 Georg Schneemayer

Abstract

This thesis proposes an architectural scheme for the implementation of contextual Web services for teaching. There are different systems that offer services for the support of traditonal teaching in universities. But until today no system exists that is flexible and powerful enough to fulfill the needs of all varieties of courses. Many of these systems offer only some of the supportive services that are needed. However there exist solutions for a multitude of services. The combination of these solutions in a proper way may result in a system as needed, a system flexible and powerful enough. The architectural scheme proposed at the end of this thesis is a frame for building such a system with existing and newly created services. The preceding chapters discuss and present relevant research areas, existing solutions and systems, and classify the possible services.

Zusammenfassung

Diese Diplomarbeit schlägt ein Architektur-Schema für die Implementierung von kontextabhängigen Web-Diensten für die Lehre vor. Es existieren verschiedene Systeme, die unterstützende Dienste für die Präsenzlehre in Universitäten anbieten. Keines dieser Systeme ist jedoch flexibel und mächtig genug, um den unterschiedlichen Anforderungen der verschiedenen Lehrveranstaltungen gerecht zu werden. Viele dieser Systeme decken nur einen Teilbereich der benötigten unterstützenden Dienste ab. Jedoch existieren für eine Vielzahl der Dienste bereits Lösungen, die, geeignet miteinander kombiniert, ein solches flexibles und mächtiges System bilden könnten. Ein Rahmen für eine solche Verknüpfung von bestehenden und neu zu schaffenden Diensten und Anwendungen ist das am Ende der Diplomarbeit vorgeschlagene Architektur-Schema. In den Kapiteln zuvor wird auf die relevanten Forschungsgebiete, die bestehenden Lösungsansätze, die existierenden Systeme sowie auf die möglichen Dienste eingegangen.

Contents

1 Introduction

The goal of this thesis is to propose the implementation of contextual Web services for teaching. Such services can support the traditional teaching in universities. Some of them may be useful for distance teaching, too. But the focal point of this thesis are supportive services that complement traditional teaching efforts by making use of services implemented with up-to-date techniques. Both the students and the teaching staff will profit from such services. The services will function as catalysts for the communication between the different participating groups and also for the intra group communication. They will allow a more efficient management of courses and a better analysis of the success of the teaching efforts. Students are supported in their development of study skills, they get information and support adapted to their individual needs.

To make this vision of services become reality, one needs to first investigate the status of today's research and development. One needs to be aware of the different research topics that are relevant, of the different solutions for the implementation of context aware services that adapt to the users' needs, and one needs to be aware of the existing work, of the systems that already exist, which implement different services needed for teaching. This knowledge constitutes the first column on which an architecture for the services can be built. The other column consists of ideas for services that are useful for teaching. Concrete proposals need to be formulated for such services to get a working and load carrying second column.

Only when this work is done, one can rise to propose a scheme for the implementation of a system for such services. An architecture that allows both the use of existing services and the development of new ones, the configuration of the services according to the needs of different forms of courses used in traditional teaching, and an architecture that enables a stepwise implementation of a usable system.

This thesis is structured as outlined above. After presenting the status of today's research and development in Chapter 2 and 3, representative existing systems are presented, described, and classified into four groups in Chapter 4. These chapters constitute the first column while Chapter 5 constitutes the second one. It uses a service centric classification in order to support the presentation of useful and needed services for teaching. With these two columns as a foundation an architectural scheme that fulfills the goals mentioned above is described in Chapter 6. This architectural scheme is an open one that allows to freely compose different teaching services, existing and new ones, in a common way step by step, forming a system of contextual Web services for teaching.

2 Overview of Relevant Areas

The introductory chapter has already mentioned that there are different areas that are of interest for contextual Web services for teaching. Three areas will be investigated in more depth in this chapter. Teaching is cooperative work, as well as learning is cooperative. So it is clear that computer support for cooperative work is one of the three areas. The other two, contextual computing and adaptive hypermedia, have to do with the question, how contextual information can be gathered, processed and made use of to adapt to the individual needs of the users.

2.1 Computer Supported Cooperative Work

Computer Supported Cooperative Work (CSCW) is the research field concerned with the design, adoption, and use of groupware. Groupware is software that supports groups in tasks like communication, information sharing, cooperative authoring, work-flow management, appointment and task scheduling, and decision making. Groupware applications are commonly classified with respect to their support of group work in the dimensions of time and place:

time: group members are working together synchronously or asynchronously

place: group members are working together co-located or at remote locations

Asynchronous and remote groupware systems can take special advantage of the Internet as a time and place independent medium.

As groupware installed in companies can support the different project teams, e.g. in the building of a software system, by providing communication tools, shared workspaces, etc., groupware can also support teams in collaborative learning. Collaborative learning can be seen as the process of shared knowledge building analogous to the building of a software system. The participating learners communicate their ideas and experiences and exchange information to discuss about knowledge in order to construct personal knowledge that serves as a basis for a shared understanding and solution of a problem. This is consistent with the constructivist view of learning, where learning is a dynamic process of knowledge construction [VVD01].

In the field of Computer Support for Collaborative Learning (CSCL) there are some recent studies that examine the application of groupware in academic, especially computer science ([GEL02], [DBR01]), education. Both studies came to the conclusion that education benefited from the introduction of groupware, but to different extent, depending on the students' self-study skills, the familiarity with the groupware system, the user base and the task the groupware is used in. For an effective use of groupware it is essential to guide the students in the beginning to make them proficient with the collaborative tools.

In the following sections, some typical features of groupware that are especially applicable to collaborative learning are presented.

2.1.1 Computer Mediated Communication

Computer Mediated Communication (CMC) covers all communication services provided by computer systems. CMC can be categorized with respect to time into asynchronous and synchronous communication and with respect to the number of peers at the endpoints of the communication:

		time	
		synchronous	asynchronous
peers	one-to-one	private chat	e-mail with only one recipient
	one-to-many		announcements, e.g. as an e-mail
	many-to-many	chat	newsgroups, mailing lists, public annotations

The asynchronous communication forms are particularly useful in Internet group work, as they bring together the group members not only independent of their location, but independent of time, too. There is no need of a coincidence in time and place for the group members to collaborate. On the other hand, synchronous communication, like a chat, to be efficient, needs a large user base to ensure that at least one suited peer is available to answer a question of another peer. Considering the above, an implementation of primarily asynchronous communication tools is more promising in the field of support for academic education.

2.1.2 Annotation

Annotations are already mentioned in the previous section about CMC, but although they can be seen as CMC, annotations are sufficiently special to justify a section of their own.

Annotations are connected to documents. The four common activities on documents ordered by decreasing pervasiveness are reading, annotations, collaboration, and authoring. While tools for reading, writing, and to a lesser extent collaboration on electronic documents are in widespread use, tools for the annotation of electronic documents are uncommon, in contrast to their widespread use with paper documents. Two different forms of annotations are possible: public and private annotations. While public annotations could be seen as another many-to-many CMC instance, private annotations are in another class and not a part of CSCW. Technically the two are very similar.

Annotation on the Web is in general only possible externally to the annotated document, because the user typically has no write access to the document he wants to annotate. But the advantage often cited of being independent of updates to the document, meaning that annotations are preserved even if the target document changes, raises a new problem: it is difficult to ensure that an annotation points to the right part of a document after the document has been modified. This problem of so-called robust annotations is subject to active research [PW00]. A solution to this problem would increase the usability of electronic annotations. Today users are faced with the problem of re-attaching orphaned annotations and are often confused by mispositioned annotations, which result from automated attempts to attach an old annotation to a modified document. Users seem to prefer orphaned annotations over mispositioned annotations [BBGC00].

An example of an implementation of annotations for Web-pages is W3C's Annotea Project. It uses XPointer to locate specific parts of documents and stores the annotations as RDF data using XML documents sent over HTTP to a server. Using Annotea requires an enhancement

of the browser itself or a browser plug-in. Though Annotea is a promising project because of its use of existing standards and its status of a W3C LEAD Project, it has currently two major flaws:

robust annotations: XPointer, which is used for locating the annotated document part, is not designed to support intra-document location in changing documents, but changing documents are very typical for the Web.

Other approaches look more promising, such as the one described in [PW00]. It uses a combination of automatically generated location descriptors and heuristics. For locating one part of the document different location descriptors exist. Each of the location descriptors captures different aspects of the document, but target the same location. The heuristics are used in case a descriptor fails to locate its target and provide a degree of confidence for the hypothesized location.

security and privacy: Storing all annotations on a server, opens a potential security problem. As widespread Internet standards are used for Annotea (HTTP, RDF, etc.), the well known security features of these standards can be used, making Annotea as secure as any other Internet services. But the problem of privacy (such as for private annotations or annotations that should only be accessible for a specific group) can not be addressed solely through secure protocols. Hosting private annotations on the client would avoid the privacy problems with private annotations. Another approach would build up on top of Annotea's protocol: encryption of the annotations. But this would only encrypt the content of an annotation, not the fact that an annotation has been made at all.

As the Annotea members are aware of these problems, some future enhancements probably will address them [W3C02a].

2.1.3 Collaborative Authoring

Collaborative Authoring can be seen as another instance of CMC and annotation, but here annotation and communication take place in the document that is also the goal of the activity. Collaborative Authoring is a direct way of collaborative knowledge building, as the shared knowledge of a specific subject is explicitly expressed in the resulting document.

Specific to collaborative authoring is some kind of version control functionality that allows the concurrent modification and later the merge of the then different branches of a document. Collaborative authoring tools profit from the availability of other groupware functionalities like CMC and annotation.

2.1.4 Group Browsing

Group Browsing means building up knowledge in a group collaboratively, either synchronously or asynchronously, while browsing and through browsing. Group annotations might be a part of this building process while browsing, but there are also other possible services. Real world browsing, like in a library, may be very rich in interactions with other people. Other people doing similar tasks and the librarians are asked for help, people learn from other people's

behavior, etc. [TNP97]. Group browsing tries to enable people doing these things when they browse on-line.

A system that supports these activities needs different services [dJHRCV01] including:

communication and sharing: These services are common to groupware systems, not special to group browsing.

awareness: Awareness means, that the systems makes users aware of events, actions and the presence of other users, supporting collaboration and situated actions. This is essential for the synchronous features of group browsing, like asking other people for help on some activities.

document classification: Support for document classification and inter-document relations enables the system to present users similar documents while browsing.

document rating system: A document rating system can be connected with the document classification system to improve the suggestion of relevant documents similar to the document currently being displayed.

2.1.5 Scheduling

Scheduling is another application of groupware. Every group needs to manage their work to reach their shared goals. Scheduling software enables the group to explicitly plan and coordinate their activities. A scheduling software can help to plan complex activities and to keep with the deadlines of the project plan. The scheduling part of a groupware system may use the CMC tools to inform the participants about a forthcoming meeting, etc. The integration of the scheduling functionality with existing tools, like tools for CMC, is vital for the successful adoption of the service, but for small groups and projects the "manual" planning of meetings and other activities is often more convenient than the usage of a scheduling system [DBR01].

2.2 Contextual Computing

Contextual Computing is about services that can discover and take advantage of contextual information. Extending the general definition of context in Merriam-Webster's Collegiate Dictionary, [GS01] define context in contextual computing as

> the interrelated (i.e. some kind of continuity in the broadest sense) conditions (i.e. circumstances such as time and location) in which something (e.g. a user, a group, an artifact) exists (e.g. presence of a user) or occurs (e.g. an action performed by a human or machine)

This extended definition contains the four dimensions of context that can be used to adapt services to the user's needs. Usually the location, the time, the environment and the user profile are distinguished:

location: The location can be a geographical location or an electronic location, which may be described by a URI. The location could be the real location or a pretended location. There are different degrees of abstraction possible: from the exact geographical location defined by longitude and latitude to abstract locations like "at the office", "in the car", etc. These locations often specify activities, such as working, driving, and are known as "Virtual Locations" [BK02c], too.

time: The time is relevant to the context in both absolute and relative manner. Absolute time is often used in a scaled format, like "working hours" or "weekends", and not as exact time. Relative time (two days before a specific event, during a specific event, etc.) is often more important to the context than the absolute time, e.g. for the implementation of a work-flow.

environment The environment describes what is available at or from the physical or electronic location, what activities are possible at the location, what are the properties of the location, and what restrictions apply to the usage of the service in the environment (e.g. the capabilities of the device used to access the service). The electronic location is also known as the computing context, the physical location as physical context.

user: The specific user gives access to second level contextual information stored in and inferred from a user model. A user model need not be specific for a single user, it is also possible that it represents a group of users sharing a context. Examples for the second level contextual information are user interests, preferences and knowledge.

An example of how such contextual information is used today in providing contextual services to a user are car navigation systems. Although they are based on a fairly simple model for the context (location: geographical location, determined by GPS in combination with street maps and measured movement; activity: driving; time: e.g. rush hour; environment: the traffic information, capabilities of the terminal in the car, user profile: e.g. preference for highways) and are usable only for a specific purpose, they provide real benefit to the user compared to traditional street maps. Other commercial applications of contextual computing are the so-called location based services, which are integrated in car navigation systems or available from mobile phones and assist the user in tasks like the reservation of a room in a nearby hotel.

As seen with these examples, contextual computing is primarily about mobile computing, but many of its concepts and ideas are also applicable in the virtual space of hypermedia applications. Also hypermedia applications are increasingly deployed to and accessed from mobile devices. Recent contextual computing applications take factors like the social environment and the interaction possibilities with co-located users into account, similar to cooperative work systems. All this makes contextual computing relevant to contextual Web services.

More versatile applications of contextual computing that are not restricted to a fixed terminal client and to a specific purpose, and take also other contextual information than the location into account, need a more complex model for adaptation. The following lists the problems that arise at the three levels of the adaptation process, which consists of context acquisition, context modeling, and utilization of context information.

context acquisition The question is how the relevant context data can be acquired. Simple systems like today's location based services require only the geographical location of the

user, which is known through the cell in which the mobile phone is located that is used to access the service, through GPS or through user input. This location data is then used in conjunction with existing systems like hotel room information systems. To take the physical location of the user into account, whether he drives a car, travels by train or is in his office, needs more information on the user's activity and environment. This information can be provided by the user explicitly or through his calendar information, or it can be gathered autonomously by the system by actively scanning the environment, leading to the problem of filtering the relevant information.

context modeling Data automatically gathered by sensors or by logs of system usage needs preprocessing in order to generate information on a layer of abstraction that is usable for and in a vocabulary that is understandable by the application. The preprocessed data needs to be merged with the acquired information from the other sources and with the preferences of the user. The TEA Project [GBS00] shows how multiple simple and easy to acquire sensor data can be combined through multiple levels to a more abstract level of context describing situations like "in a meeting". As already mentioned, these virtual locations are useful, because they may describe activities.

utilization of data Also the question how and to what extent the context information can be used to provide the user an optimal service is important. How can the presentation be adapted to the context, is it possible to predict the intention of the user and suggest him the next step to achieve his goal? How can the data be reused by other services, can different services (perhaps accessed from different devices) share a common user model without neglecting security and privacy concerns of the user?

As these problems show, contextual computing needs further active research to reach its goal of extending systems with abilities to sense contextual data, to represent it, and to work with it for the sake of a better support of the users' needs through adapting the behavior of applications.

2.3 Adaptive Hypermedia

In this section adaptive hypermedia is further investigated, a research area related with contextual computing, but with its roots in hypermedia and user modeling. It is concerned with the adaptation of hypermedia content and links with respect to the context. Web-based hypermedia systems offer access to a great amount of information and services. The Web offers the user a large amount of navigational freedom and information resources. The downside of this freedom in traditional hypermedia systems is expressed in the terms "lost in hyperspace" and "information flood", the terms show the need for assistance and guidance in the Web [Ham89]. Moreover the classes of users to be dealt with in Web applications are very heterogeneous due to the worldwide accessibility. Typical Web applications like information systems, learning systems, or electronic commerce systems will benefit from adaptive hypermedia systems. With adaptive hypermedia they have an alternative to the traditional "one-size-fits-all" [Bru01] approach.

The term "adaptive" needs a definition for this text, because it is used with different meanings in the literature. In this text "adaptive" means that the steps of the adaptation process (initiation, proposal, decision, execution [DMKSH89]) are executed by the systems autonomously

without user interaction. Some authors do not differentiate between user driven adaptation and autonomous adaptation executed by the system. Instead they differentiate between adaptation at runtime and adaptation before runtime. Adaptation that occurs at runtime is called there "adaptive", whether with or without user interaction and control. Adaptation that occurs before runtime is called there "adaptable" [SPS+98]. This latter kind of adaptation is called "configurable" or "customizable" by other authors.

2.3.1 Steps of the Adaptation Process

As with contextual computing, the three steps of the adaptation process consist of acquisition, modeling and utilization as seen in Figure 1.

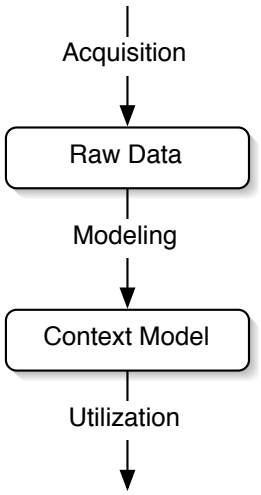

Figure 1: Three Steps of the Adaptation Process (based upon [Bru96])

2.3.2 Acquisition and Modeling

The acquisition of the raw contextual data consists of simply logging the user's activity. This is done automatically by both the server that delivers the hypermedia resource and the client that is used to access it. But as with raw sensor data in contextual computing, raw logging data is not usable with adaptive hypermedia systems. It needs to be converted and merged with other sources to the different categories that are distinguished in the research area of adaptive hypermedia. In adaptive hypermedia another categorization of contextual data is usually used than in contextual computing. The common ([KKP01], [Bru01]) categorization is to distinguish between user data, usage data and environment data.

user data: User data describes personal characteristics and preferences of the user. It can be sorted in the following categories: user's goals and tasks, knowledge, background (includes also demographic data), hyperspace experience, preferences, user (long time) interests and user's individual traits (e.g. learning style).

usage data: Usage data consists of data about the user's usage of and interaction with the system. It can be categorized in observable usage (selective actions, temporal viewing behavior, ratings, confirmatory or dis-confirmatory actions) and inferenced data like usage frequency, situation-action correlation and action sequences. Usage data is short time data, often only valid during one single browsing session. Data that is valid for a longer period, becomes part of the user data, e.g. part of the user's interests.

environment data: Environment data is data about the user's software and hardware environment and the location of the user. This category has its counterparts in the categories location, time and environment in contextual computing.

There are various methods to reach such an abstract model from the raw data, including explicitly supplied information of the user and complex user modeling systems with acquisition rules, plan recognition, stereotype reasoning and deductive reasoning capabilities [KKP01].

To get a single data model for user model adaptation from the multiple sources without data loss, often a so-called overlay model is used. In the overlay model the information from the different sources is stored in different layers, each data source on its own. Then the layers are simply combined by laying one layer upon the other where the upper layer overwrites conflicting properties which are already defined in lower layers. This results in a single model, but it is only applicable if a strict precedence order of the different data sources exists. More sophisticated overlay models are based on weighted layers. Thus they are not in the need of a strict precedence order of the data sources.

Another problem of the acquisition of user data are privacy concerns of users as well as the privacy laws of various countries [Kob02]. The storage of the raw usage data beyond one single session as well as the use of this data for various systems is very problematic. A solution to this problem is to use pseudonyms for the users. This means that the component that receives data from identifiable users is isolated from the usage data of the user modeling system. The two components are only allowed to communicate with a third trusted component as intermediatary. If such pseudonyms are used for user modeling, privacy laws will no longer apply and the privacy concerns of the users will be reduced.

2.3.3 Utilization

The modeled data is used for adaptive presentation and adaptive navigation. Figure 2 shows a taxonomy of the different types of adaptation that are possible.

Adaptation of Content The first main branch of adaptive hypermedia technologies in this figure is the adaptation of content. Adaptation of content is concerned with the question, what content should be presented to the user. In the following some methods of the subtree adaptive text presentation are described:

Fragment Display: Fragment display means, whether or not to display specific fragments of a page. The fragments can exclude each other as in fragment variants or they can be additive as in stretch-text. It is used in order to give optional explanations or more detailed information or, in e-commerce applications, personalized recommendations for products the user might be interested in.

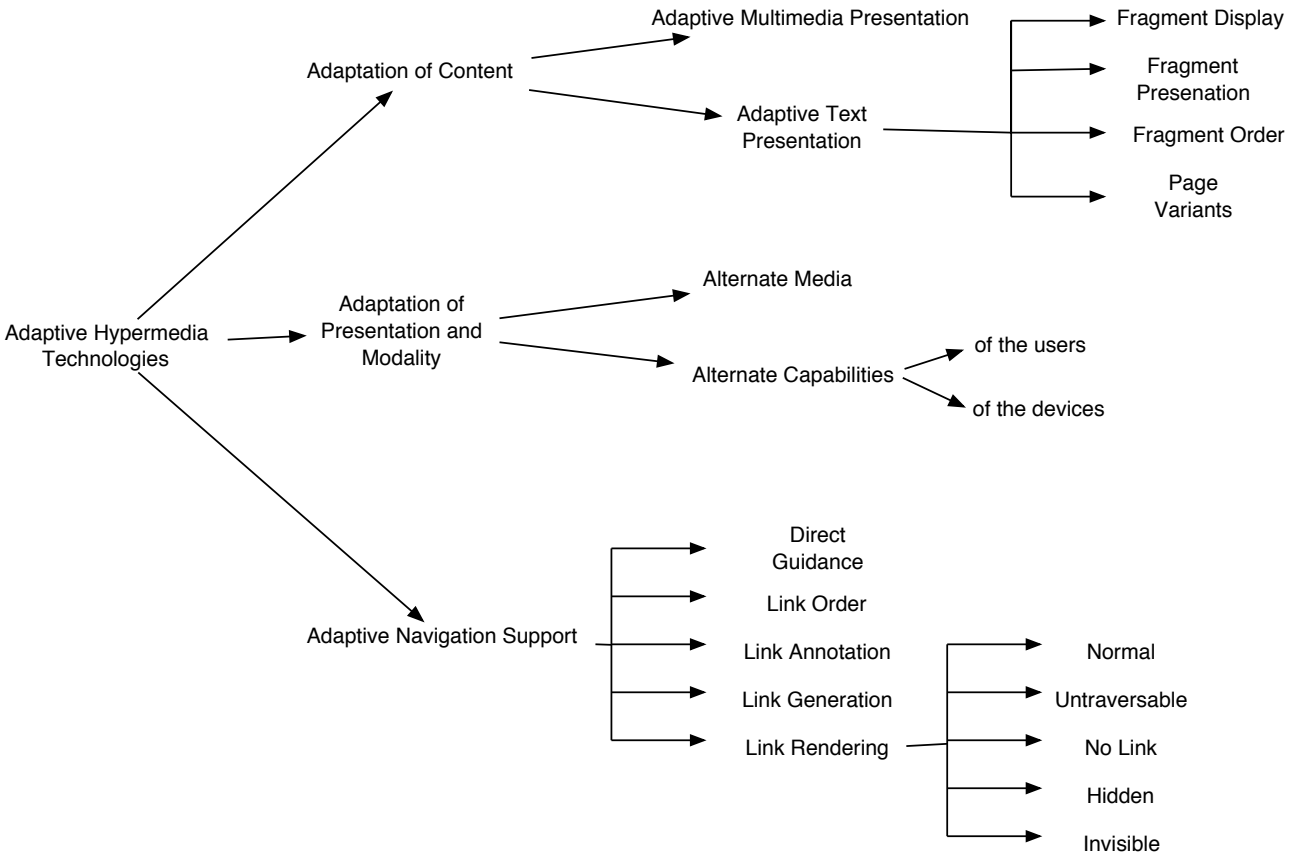

Figure 2: Taxonomy of Adaptive Hypermedia Technologies (based upon [Bru01] and [KKP01])

Fragment Presentation: Fragment presentation means different presentation variants for fragments of a page, examples are fragment coloring or dimming fragments. It is used for the same purpose as fragment display alternatives, but with the difference, that errors in the user model concerning the goals, interests, or knowledge level of the user are less fatal. All fragments are presented to the user, also the ones that the system classified as not-relevant to the user's context. The downside of this approach is, that it is only applicable, if the content can be presented in the same formulation to all users and where the variability of the content is relatively low.

Page Variants: Page variants means that different pre-defined versions of a page exist from which at runtime the suitable one is selected. This approach is from a technical point of view a very simple one, but it is also a very inflexible one. It can be used only if there are predefined types of users or usage scenarios.

Adaptation of Presentation and Modality The second main branch, adaptation of presentation and modality is concerned with the question how the content should be presented, in which format and in which layout.

Alternate Media: Adaptation to alternate media means the adaptation to different output medias, like screen, printer and aural devices.

Device Capabilities: Adaptation to device capabilities includes the adaptation to output (number of colors, screen size, etc.) and input (keyboard, handwriting or speech recognition, touch screen, etc.) device capabilities.

User Abilities: Adaptation to user abilities means adaptation to the physical abilities of the user. E.g., the modality of the presented media can be changed from image to voice for blind people.

Adaptive Navigation Support Adaptive navigation support as the last main branch is concerned with the two questions where to link to and how to link. The following describes some methods for this purpose:

Direct Guidance: Direct guidance can be helpful in familiarizing first-time users with a site, improving the guided site tours that can be found today on many sites. Another application of direct guidance are tutoring systems, where the user is guided to the learning material according to his knowledge.

Link Rendering: Combining different descriptions of link rendering possibilities of various authors ([BK02c], [KKP01], [Bru01]), five different presentation facets can be identified:

Normal: The link is traversable and rendered as hyperlink.

Untraversable: The link is not traversable, but rendered as hyperlink.

Hiding: The link is traversable, but not rendered as hyperlink.

No Link: The link is not traversable and rendered as normal text.

Invisible: The link is not traversable and is not rendered at all.

These different styles of link rendering can be used for various intentions. Examples include a reduction of the hypermedia space by making links to non-relevant information or information that is not yet appropriate to the learner invisible, and an introduction of a relevance order for links by hiding less relevant links.

Different techniques were used up-today to enable these adaptation methods. AHA! ([AHA02]) uses an enhancement of XHTML with special tags for expressing the conditions and alternatives. The enhanced XHTML pages are then filtered on the server side to standard XHTML. Other methods rely on domain specific formats and external condition specifications (e.g. OMdoc [Koh00] and ActiveMath [MAB$^+$01]).

3 Essential Components of Systems Providing Teaching Services

In this chapter some concepts, methods and techniques to implement essential components that are needed for contextual Web services for teaching are presented. Figure 3 gives an initial overview of the core components of an adaptive system. It is a conceptual view of an

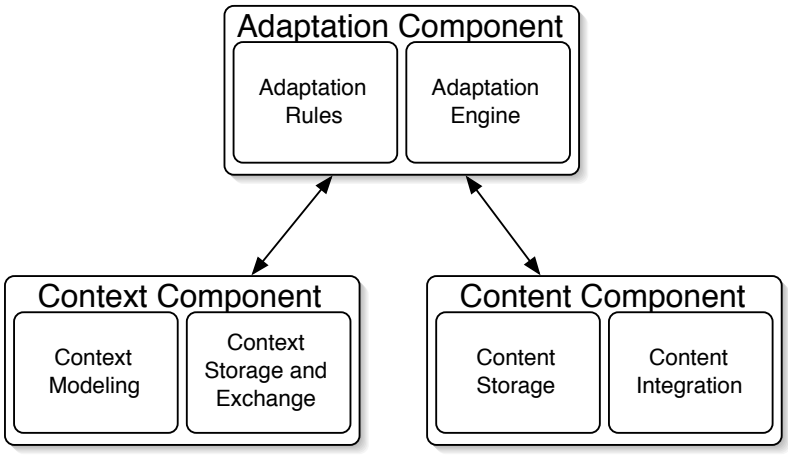

Figure 3: Core Components of Adaptive Systems

adaptive system, the different components often do not exist in this form in real systems. Some of the components may be merged and some functionalities of the components may differ. The conceptual view presented here is based on different architectures ([SO00], [MAB+01], [AHA02]), models ([BHW99]) and component implementations ([HL02], [Kob01]).

In the following sections the responsibilities of these core components are explained first. After this some concepts, methods and techniques for their realization are presented.

3.1 Context Component

The main responsibilities of the context component during the adaptation process are the storage and the modeling of contextual information. For these purposes the context component has to retrieve at the beginning of a session an initial set of contextual information as a basis for later updates. This initial set may consist of a predefined generic set, a user provided set, or an already stored set for a specific user generated in a previous session. After this initial retrieval updates to the contextual model are only made through the adaptation component in response to user actions. Exceptions may be nomadic systems, where the location of the user can change without direct interaction with the system, or systems with a scheduling component, where events may become relevant to the context of the user during time.

3.1.1 User Modeling Systems

Implementations of such context component systems are so-called user modeling systems or servers. The development of separate user model systems is part of the ongoing move to component-based architectures for adaptive hypermedia systems. There exist both research and commercial user modeling systems [Kob01]. As they differ in both functionality and architecture they are described in the following in own sections.

Research Prototype Systems Most of the research user modeling systems were developed in the first half of the nineties. They tend to be very general with respect to domain independence, expressiveness, and strong inferential capabilities. The reason for this is that user modeling research in those days had an affinity to artificial intelligence, natural language dialog, and intelligent tutoring. A representative of such a system is BGP-MS (Belief, Goal and Plan Maintenance System) [KP95]. BGP-MS represents assumptions about a user and stereotypical assumptions about a group of users in first-order predicate logic. Inferences in BGP-MS are based on messages from the application and from previously made assumptions. A modal logic is used to represent the different assumption types. The application developer may also define LISP functions to be called on specific messages. BGP-MS is implemented in Common Lisp and runs as a separate server process independent of possibly concurrent client applications.

Although research systems like BGP-MS have advanced modeling and sometimes also client-server capabilities, they did not reach wide distribution, neither in research nor in commercial environments. Reasons are, that today's user-adapted applications like learning environments and user-tailored Web sites are domains with less demanding user modeling requirements.

Commercial Systems Commercial user modeling systems do not have such advanced user modeling capabilities like the research systems. They are instead more concerned about a scalable and reliable architecture. The modeling techniques are relatively simple ones like stereotype reasoning and collaborative filtering. These techniques allow another feature important for commercial systems: quick adaptation. This is required in order to bind users by providing adaptation already for first-time visitors. For this purpose some systems can switch between different modeling and personalization methods, depending on the amount of acquired data. Another feature that most research prototype systems lack completely is support for privacy. Various studies referred by [Kob02] show that Internet users are (very) concerned about privacy threats (between 81% and 87%) and being tracked (between 54% and 77%) when using the Internet. In order to respect these user demands and also to conform with the privacy laws of various countries, support for privacy is required by non-research adaptive hypermedia systems. A way to implement this privacy support was already presented at the end of Section 2.3.2.

3.1.2 Context Storage, Exchange and Update

The storage format of the contextual information is another important aspect, especially if the contextual information should be reused by different applications, devices or users. The early user modeling servers had no means for exporting and importing contextual information

other than with proprietary interfaces. For commercial implementations at least importing capabilities for integrating customer and marketing data from existing databases are a must. But a reuse and exchange of contextual information needs more advanced techniques than a simple data import mechanism. Also a mechanism for updates to the context information is needed. Applications that want to take the behavior of the user into account, need to update the context information with usage data.

In the following two standards for the exchange and storage of contextual information are presented, W3C's Composite Capabilities/ Preferences Profiles [W3C00] and IMS Project's Learner Information Package Specification [IMS01b]. After that a more lightweight approach for the storage of context information on the client side is presented which is in contrast to the previous approaches aware of the need for update-able context information. This last approach is also concerned with adaptation techniques, making it also relevant in Section 3.3, the section about the adaptation component.

3.1.3 Composite Capabilities/Preference Profiles

Composite Capability/Preference Profile (CC/PP) is a framework for the description of the users' device capabilities. There exist proposals of using CC/PP for other contextual information like user preferences or calendaring and scheduling information, too [HN00].

The description of the contextual information of CC/PP is represented in RDF. RDF is designed as a machine readable general purpose meta-data description language. The RDF model can be represented in different XML serializations, depending on whether or not elements or attributes are used to represent properties and other factors. It must be mentioned that an XML serialization is only one possible serialization of CC/PP. CC/PP is independent of a specific XML serialization. The following explanation of CC/PP must be read with this in mind. This means, that the terms used with CC/PP are sometimes the same as the terms used with XML. This does not mean that the terms have the same semantics.

A CC/PP profile consists of a two level hierarchy: components and attributes of components. Attributes can be simple (one-value) or complex (multiple value). The attributes of a component can be specified in place or can be referenced. It is also possible to reference first a default profile and then to specify differences in place. The in place values take precedence over the referenced ones in both cases. Example 1 shows an XML serialization of a small profile with a "Hardware Platform" component which references Example 2 for the default attribute values. The in place "memory" attribute with the value of 32Mb overwrites in this example the referenced value of 16Mb.

The CC/PP recommendations do not contain a specific set of vocabulary of capabilities and preferences, but instead choose to defer this definition to other standards bodies. The WAP-Forum is such a standards body that chose to make usage of the CC/PP work of W3C and developed a vocabulary applicable for mobile devices and WAP 1.1 called UAProf (User Agent Profile, [For01]). The examples 1 and 2 use this UAProf vocabulary. The HardwarePlatform component is one of six different components defined by UAProf. Other components allow for example the definition of software and network capabilities and user preferences.

For the transmission of the profile the CC/PP Working Group has proposed a protocol using the HTTP Extension Framework (HTTP-ex), which itself is an incompatible extension to

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#">

  <rdf:Description rdf:about="MyProfile">

    <ccpp:component>
      <rdf:Description rdf:about="TerminalHardware">
        <rdf:type rdf:resource="HardwarePlatform" />
        <ccpp:defaults rdf:resource="HWDefault" />
        <memory>32Mb</memory>
      </rdf:Description>
    </ccpp:component>

  </rdf:Description>
</rdf:RDF>
```

Example 1: Device profile referencing defaults

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#">

  <rdf:Description rdf:about="HWDefault">
    <rdf:type rdf:resource="HardwarePlatform" />
    <display>320x200</display>
    <memory>16Mb</memory>
  </rdf:Description>
</rdf:RDF>
```

Example 2: Defaults for HardwarePlatform

the HTTP 1.1 protocol. Because of the incompatibility of HTTP-ex to current Web-Servers implementing HTTP 1.1 and the confusion about the status of the protocol [W3C02d], the UAProf Working Group uses an own backward compatible extension to HTTP 1.1 called Wireless profiled HTTP (W-HTTP). Both protocols have essentially the same features and so they are described together in the following. The CC/PP profile is associated by the client with each request to the server. Profile references and profile defaults can be used with both protocols. This allows the storage of a default profile on a third device called profile repository resulting in smaller CC/PP profiles (which are especially suitable for mobile devices) and in less bandwidth usage (which is especially suitable for mobile devices, too). This also allows a manufacturer of an access device to keep the device information always up to date. A user with different devices benefit from this feature, too. The personal preferences of such a user could be stored on a server and referenced by the different devices of the user. This eliminates the danger of data inconsistency that comes with redundant storage and unburdens the user from keeping different user profiles on different devices up to date. The composition of the explicit and referred parts of the CC/PP profile is called profile resolution. The server application is responsible for this resolution.

Because the CC/PP profile is sent by the client to the server, every intermediate gate has access to the CC/PP profile. From a security and privacy point of view this is no good thing, but it also has a powerful use case: a mobile device with only limited display capabilities that accesses a Web page through a gateway. The gateway can act as an enabler for the mobile device, it can adapt the content of a Web page that has no or not sufficient adaptive features to the capabilities of the client. This so-called in-band content transformation changes both the CC/PP profile of the client (changing the client capabilities to the gateway capabilities of adapting third party content to the original client capabilities) and the content delivered by the server (according to the original client capabilities).

There already exists a toolkit for CC/PP that can be integrated with Apache Group's publishing framework Cocoon allowing the delivery of content according to the CC/PP profile of the client: DELI [HL02]. DELI is described in more detail in Section 3.3.2.

3.1.4 Learner Information Package Specification

The IMS Learner Information Package Specification [IMS01b] defines a data model of the characteristics of a learner. The specification addresses the interoperability of Internet-based "Learner Information systems" with other systems that support the Internet learning environment. "Learner Information systems" as used by IMS can be compared to some extent with user model servers. They do not have any modeling capabilities, but allow the centralized or distributed on-line storage and retrieval of a user model.

The user model defined by the specification allows the modeling of both learning history and learning objectives. Both short time learning goals and life long learning records can be stored in the user model. To achieve this, a composite approach is undertaken, where only the required information needs to be stored. The most interesting core data structures for the purpose of adaptive systems are:

Identification: biographic and demographic data that is relevant to learning

Accessibility: language proficiency, disabilities, eligibilities, learning preferences, physical

preferences (e.g. preferences for large print), and technology preferences (e.g. computing platform)

Goal: learning goals, career goals; a further division into subgoals is possible

Competency: skills, knowledge, and abilities already acquired

For every field in the data model meta-data for time-related information and privacy information is applicable, either directly or via inheritance.

Even though the interoperability of learning systems and so the exchange of learner models is the scope of the specifications, up-today the exchange and query of learner information is stated to be out of scope. Only some basic requirements like transaction management and a recommendation for the aggregate packaging of multiple information instances [IMS01a] are mentioned.

IMS Learner Information Packaging Specification seams to be more appropriate for application areas which require a finer granularity of user and context information than adaptive hypermedia systems. Nevertheless adaptive hypermedia systems can profit from developments and specifications that are and will be made with the IMS Learner Information Packaging Specification in mind, like exchange protocols and privacy implementations.

3.1.5 Style Sheet Selectors

Style sheet selectors ([BK02b], [BK02a]) for adaptation is an approach that uses existing standards with small extensions for making them ready for adaptation functionalities. Style sheet selectors use path selectors as found in CSS and XSLT for content and navigation adaptation. For this purpose the approach suggests to enhance the browsed XML and HTML documents with an XML data structure for expressing the context information called "browsing context" on the client side. The result is a "content enriched document". This enables CSS and XSLT style sheets to take the browsing context into account for the presentation of the original document. Web applications can update the browsing context using scripting languages like Java-Script, which also have access to the DOM of the content enriched document. Parts of the browsing context can be updated by the browser itself. It is not the intent of the proposal to define a data format for the browsing context, but some suggestions are made how the browsing context could look like to be useful for adaptation.

The browsing context, as it is proposed, consists of three kinds of information: browsing history data, browsing environment data and application data. The browsing history data is updated automatically by the browser and consists of the past browsing actions of the user, first of all the accessed Web pages. The browsing environment data consists of the device capabilities, location, time, language, etc. The information contained corresponds to the information that can be specified by UAProf and CC/PP. The third and most versatile kind of information is the so-called application data. The application data is in contrast to the previous kinds of data not stored and managed by the browser, but by the individual Web application that is currently used. It is specific to the individual application, for a tutoring system it could store information about the learner's performance in exercises, for an e-commerce application the content of the shopping cart. Example 3 shows an outline of a context enriched document with an HTML document as original document.

```
<root>
  <browsingcontext xmlns="http://www.example.com/browsingcontext">
    <!-- browsing history data  -->
    <webpage uri="http://www.example.com/index.html">
      <webpage uri="http://www.example.com/browsingcontext/index.html"/>
    </webpage>

    <!-- browsing environment data  -->
    ...

    <!-- application data -->
    ...

  </browsingcontext>

  <html xmlns="http://www.w3.org/1999/xhtml">
    <!-- head and body of the original html document -->
  </html>
</root>
```

Example 3: Outline of a Context Enriched Document

The proposal also suggests a small extension to the current style sheet selectors of CSS and XSLT: the preposition of context document selectors, making the original document selectors context conditional. Selectors using CSS and XPath as XSLT's selector language are shown in examples 4 and 5. The examples hide or render a link to a page with first time visitor information, depending on whether the user already visited it or not.

```
bc:webpage[uri="http://www.example.com/first-time-visitor.html")]
    html:a.firsttime {display:none}
```

Example 4: Context-Enhanced CSS Selectors

The matching algorithm can be left almost unchanged with this extension. Also this extension is backward compatible: processors incapable of handling browsing contexts that try to match these enhanced selectors will never succeed. Effectively they will simply ignore the context dependent rules.

The proposal is very interesting in respect of two things: the relocation of the adaptation process to the client side and the need for only small changes to existing standards resulting in powerful adaptation possibilities. The advantages of the client side processing are the possibility to enhance the user's privacy and the unburdening of the server from the expensive content transformation. The backward compatibility of the proposal reduces the disadvantage of the need for browsing context capable client. Non-capable clients can further on access context enhanced Web pages, but without the advantageous adaptive features. Client side

```
<xsl:choose>
  <xsl:when test=
      "//bc:webpage[uri='http://www.example.com/first-time-visitor.html']">
    <a href="/first-time-visitor.html">
      Information for First Time Visitors
    </a>
    <xsl:apply-templates/>
  </xsl:when>
  <xsl:otherwise>
    <xsl:apply-templates/>
  </xsl:otherwise>
</xsl:choose>
```

Example 5: Context-Enhanced XPath Selectors

adaptation with style sheets is capable of content selection, not only of content transformation. With the help of index pages that link to the actual content, the decision which parts should be presented to the user can take place on the client side. This makes client side adaptation sufficient for more sophisticated adaptive applications like learning systems, too. But there is also one disadvantage of the client side approach. Especially mobile devices have only a limited and costly bandwidth. The use of index pages that refer to the actual content lessens the transfer volume in comparison to an implementation without such index pages. But the transfer volume remains a problem of client side adaptation. As a result, this proposal may be not the best choice for mobile and other devices with high bandwidth costs.

3.2 Content Component

The content component can be described as the storage layer for the domain model and as integration layer for external content. Its main objective is to provide the adaptation layer with content in a form that is suitable for the adaptation technique used. Content that is already in an appropriate format (e.g. LOM, described in Section 3.2.1, or OMDoc [Koh00]) can be simply forwarded to the adaptation layer, but legacy content needs to be wrapped or transformed to be usable by the adaptation component. The integration of external content in a way that the adaptive hypermedia system is able to conceive the semantics of the content and the links of a document is the goal of so-called open corpus adaptive hypermedia systems. There exist already some approaches [HN02] that allow e.g. the integration of learning content with standard meta-data like Learning Objects Meta-data.

The following section describes a meta-data model, which is currently under development, for the representation of learning content is further described. After this the approach of the definition of a specific document format for learning content is discussed.

3.2.1 Meta-data Approach: Learning Objects Meta-data

Learning Objects Meta-data (LOM, [IEE02]) is a standard under development by the IEEE Learning Technology Standards Committee. LOM shall facilitate the search, evaluation, acquisition, and use of learning objects. Learning objects as defined by this standard are entities, digital or non-digital, that may be used for learning, education and training. The learning objects should be usable for both humans and software services. This will enable the sharing and exchange of learning objects.

To achieve these goals, the standard specifies a conceptual data schema of meta-data for learning objects. It shall be a base schema, which can be used to build on for advanced applications like adaptive software systems. It uses existing standards like Dublin Core, ISBN, and ISO country codes for the data format where applicable. The IMS Learning Resource Meta-Data Information Model uses LOM as a basis and also developed a representation of LOM in XML. The data schema of LOM can be seen as a hierarchy of elements. The following lists the elements of the first level after the "lom" root [IMS02]:

general Groups information describing meta data as a whole. Sub-elements include identifier, title and description.

lifecycle Version information of the resource

metametadata Features of the description rather than the resource

technical Technical features of the resource like format (mime type or non-digital), size and platform requirements.

educational Educational and pedagogical features of the resource like context, difficulty and type of the learning resource (e.g. exercise, questionnaire, narrative text).

rights Usage conditions like copyright information.

relation Describes the nature of the relationship between the current and a referenced resource. Examples for kinds of relationship are "IsBasedOn", "Requires" and "IsPartOf", corresponding to the Dublin Core element "DC.Relation".

annotation Comments on the educational use of the resource.

classification Description of the characteristics of the resource by entries in a taxonomy through a ordered list of taxons.

LOM's primary purpose is not the use in adaptive systems but the more general exchange and classification of learning objects. Therefore the meta data as specified by LOM is possibly not fine grained enough for adaptive systems. Nevertheless it gives an outline how such meta data could look like and can work as a basis for a more detailed description of learning objects. LOM explicitly allows the extension with more specific meta data for such purposes.

3.2.2 Specific Document Format Approach: OMDoc and MMiSS

While LOM is a meta data schema which describes a learning object, data schemes designed for adaptive systems are often document formats which define how a learning object is

structured. Examples are the document format under development for MMiSS (described in Section 4.4.1) which is based on OMDoc. A learning object in such documents consists of one or more sections with definitions, paragraphs, theorems and so on. This approach seems more straightforward and simpler than the meta data one. But from the point of view of adaptive systems, the document format of a learning object is not relevant, relevant are meta informations on the learning object as a whole and, that is what LOM lacks, meta informations on fragments of the learning object. The document format is only of secondary interest, it is only needed for the purpose of the presentation of the document to the user. Defining and using a document format for adaptation has the disadvantage of being specific for a purpose, in the case of OMDoc specific to mathematical documents. A document format dependent adaptation component can not be used with another document format, but an adaptation component dependent on a document's meta data can be used with every document that can be enriched with the meta data.

3.3 Adaptation Component

The adaptation component is responsible for the adaptation of the content received from the content component according to the context retrieved from the context component. For this adaptation to take place, rules and conditions and a technique for the adaptation of the content is needed. The conditions can be very simple and static, as with Media Queries in CSS3 [W3C02b], or more advanced as with XSLT transformations with XPath expressions used by DELI. The conditions for the adaptation may be generated dynamically with the usage of pedagogical rules (in learning systems) or more generic adaptation rules, too.

In the following two techniques for adaptive rendering of content are presented, the simple, but often sufficient Media Queries of CSS3 and DELI in conjunction with CC/PP and XSLT. Another technique was already presented in Section 3.1 about the context component: style sheet selectors, described in Section 3.1.5.

3.3.1 Media Queries in CSS3

Device independence is especially vital for mobile devices because they come in a wide variety of input, output, hardware, software and network capabilities. To adapt to this rich variety of devices one needs to know the capabilities of the devices. A simple but for many applications sufficient approach are media queries proposed as part of the W3C CSS3 recommendation. XML documents' reference mechanism for style-sheets is extended with an additional media attribute that holds a predefined media type (screen, print, aural, etc.) and series of boolean expressions consisting of media features like color or max-height (of the screen). The boolean expression can be combined with AND, OR and NOT to more complex queries as seen in example 6.

Because these queries are part of an attribute of the element that links the style-sheet to the document, these queries are also part of the document and are interpreted at client side. The benefit of client side evaluation are privacy concerns and lower server load. Because privacy concerns are not critical when adapting only to client capabilities, here client side interpretation has no user benefit. Another result of this client side computation is, that all data of the document that is possibly needed for the presentation on any device has to

```
<link rel="stylesheet" href="css/bigcolorscreen.css"
      media="screen and (color) and (min-device-width: 1024px)
             and (min-device-height: 768px)"/>
```

Example 6: Media Queries

be transfered to the client. E.g., if the document should be rendered on a small screen like that of a mobile device, probably only an outline of the document will be rendered by the style-sheet, but the whole document has to be transfered to the mobile device because the same document needs to be able to adapt to a device with a bigger screen, like a desktop PC, as well. This is even more disadvantageous because mobile devices have typically only a low bandwidth and also high bandwidth costs. Another disadvantage of media queries is the fixed and monolithic vocabulary used. It is defined in the recommendation of CSS3 and can not be extended independently (as with the usage of name-spaces) externally. The simple and thus not very powerful syntax makes the approach of media queries not applicable for more advanced contextual descriptions, it can not be parsed and made use of with standard XML techniques like XPath and XSLT as well.

To sum up, one can say that CSS3 media queries fulfill their role as simple technique for basic device independence but are not capable in the support of more sophisticated contextual adaptation.

3.3.2 DELI

DELI is an open-source library that allows server applications to access CC/PP information included in W-HTTP requests. DELI itself only does the profile resolution and allows the server application to access the resulting CC/PP profile. The resolution consists of the merging of the explicitly given profile information, the differences, the default and the referenced profiles according to the resolution rules. How the application uses this profile for adaptation is not in the scope of DELI. There exists an integration of DELI into Cocoon, a publishing framework. With Cocoon DELI can be used to adapt content through XSLT transformations which can query profile attributes using XPath. Another approach of content adaptation with DELI and Cocoon uses so-called capability classes to match CC/PP profiles. In the following the two approaches are further investigated:

Content Transformation using XSLT DELI can make the client's CC/PP profile available for XSLT style-sheets used to process content accessed by the client through the Cocoon framework. For this purpose the XML serialization of the CC/PP profile is preprocessed by DELI to allow more simple XPath expressions in the XSLT style-sheets. This simplification of the original serialization of the profile does the following: the opening an closing tags of the component elements are omitted, all references are resolved, and also the elements in the RDF namespace are omitted or replaced by elements containing only a small part of the original information. The result is a document without different namespaces and with all properties at a nesting depth of only three from the document root. This preprocessing and simplification is called "flattening" by the DELI authors. Then the resulting document is

passed to the XSLT style-sheet as parameter. Hence the attributes can be queried by XPath expressions in the style-sheet. Example 7 shows a fragment of a style sheet, which uses this approach to distinguish between different screen sizes and the color capabilities of the device:

```
<xsl:param name="deli-capabilities"/>

<xsl:template match="/">
  <xsl:choose>
    <xsl:when test="
      (number(substring-before(
          $deli-capabilities/browser/ScreenSize, 'x')) > 320) and
      (number(substring-after(
          $deli-capabilities/browser/ScreenSize, 'x')) > 160) and
      ($deli-capabilities/browser/IsColorCapable = 'Yes')
    ">
      <xsl:apply-templates select="." mode="bigcolor"/>
    </xsl:when>
    <xsl:when test="
      ($deli-capabilities/browser/ScreenSize = '320x160') and
      ($deli-capabilities/browser/IsColorCapable = 'Yes')
    ">
      <xsl:apply-templates select="." mode="mediumcolor"/>
    </xsl:when>
    ...
  </xsl:choose>
</xsl:template>
```

Example 7: XSLT Style Sheet Using CC/PP Profile Provided by DELI

As this example shows, creating more complex constraints would be very work intensive. The flattening of the original CC/PP profile serialization only leads to shorter path selectors in the XPath expressions, but for complex constraints both the number and the complexity of each XPath expression rise. The complex expressions are the result of the need to evaluate all properties that should be taken into account in the style-sheet itself. Also every single part of the expression could become very complex, especially because some data-types of UAProf like dimension (e.g. 320x160) are difficult to handle with XPath functions. A level of abstraction to handle the different capabilities would be desirable. Also the usage of CC/PP profiles in the transformation of content is only one possible usage scenario. The CC/PP profile could be used at an earlier stage, e.g. during content creation. The described integration of DELI in Cocoon only addresses the content transformation stage. The capability classes described in the next section do not have these shortcomings.

Content Adaptation using Capability Classes As mentioned before, capability classes [But02] addresses the shortcomings of the content transformation approach with XSLT: expendable XPath expressions and the limitation to content transformation. The core idea of

capability classes is the introduction of a level of abstraction between the very detailed and complex context information and the manageable number of adaptation possibilities. This is done through the definition of capability classes in a file which uses its own syntax for defining the classes through boolean expressions and an interpreter for such a file which is aware of the different UAProf data-types. Example 8 shows a fragment of such a capability class definition file.

```
<classes>
  <class name="bigcolor">
    <and>
      <greaterthan value="320x160">ScreenSize</greaterthan>
      <true>IsColorCapable</true>
    </and>
  </class>
  <class name="mediumcolor">
    <and>
      <equals value="320x160">ScreenSize</equals>
      <true>IsColorCapable</true>
    </and>
  </class>
  ...
</classes>
```

Example 8: Capability Class Definition File

The capability classes that match are passed to the XSLT style-sheet as content of the "capabilities" parameter, analogous to the content transformation approach in Section 3.3.2, but now with the easier to handle list of matching capability classes and not with the detailed CC/PP profile tree. In addition to the availability of the capability classes in the content transformation stage, the integration of capability classes in Cocoon, which is currently under development, also allows the usage of the classes in the content selection and generation stage.

The downside of capability classes is that they introduce another language to define the classes in addition to XSLT, XPath, and UAProf's implementation of CC/PP. If the language and its interpreter should be used with another implementation of CC/PP than UAProf, both need to be adjusted to the new data-types and vocabulary. If the CC/PP profile is not only used to adapt to device capabilities but also to other context information like usage, the exact context information may be needed for adaptation, classes may be not precise enough to express the past actions of a user. Nevertheless the core idea of the capability classes, the introduction of an abstraction level to the context information and the usage of the context information in other stages besides content transformation could be useful for many adaptive hypermedia applications.

4 Survey of Existing Systems providing Teaching Services

There is a diversity of systems providing teaching services resulting from different backgrounds in research, actual necessities and other circumstances. In the following these systems are roughly classified in three groups and some representative and interesting systems are further investigated. After this classification a further project is presented, which tries to combine some projects from different groups.

4.1 Adaptive and Intelligent Web-based Educational Systems

Adaptive and Intelligent Web-based Educational Systems (AIWBES or W-AIES) are an active research topic in the field of Web-based educational systems. AIWBES combine different research topics like adaptive hypermedia, artificial intelligence and adult learning theories. They offer intelligent and adaptive support to the learner, resulting in a personalized user specific learning environment. This is especially useful in a Web environment, where it is not likely to have homogeneous classes of users. Users with different skills and different knowledge levels will use the system, expecting a suited learning environment. Another aspect is that in contrast to classroom courses, where direct interaction among students and between students and teachers is possible, there is a need for communication and collaboration facilities through the system letting the users interact with each other.

The following features are used in AIWBES in different combinations and peculiarities to achieve these objectives:

adaptive guidance and presentation: Adaptive guidance means providing the student an optimal navigation path through the learning material according to his personal preferences, goals, navigation history and/or tested knowledge.

There are different forms of guidance implemented in the existing systems, examples are adaptive link annotation or adaptive link hiding.

Adaptive presentation means the adaptation of the content itself according to the same information as with adaptive guidance.

There are different forms of adaptive guidance and presentation implemented in the existing systems. Techniques used for adaptive guidance include adaptive link annotation or adaptive link hiding. Adaptive presentation uses techniques like conditional text and stretch text. In Section 2.3.3 a more complete overview of the different techniques that can be made use of is given.

adaptive collaboration: Collaboration features are public annotations, discussion forums, chats, etc. Adaptive implementations of collaboration uses the system's knowledge about its users to classify them. This enables, among others, the generation of collaboration groups of users with matching learning goals or finding the most competent user for answering a question about a specific learning topic. This analysis of the user models also allows the tutor to identify users lagging behind in their learning success.

interactive problem solving support: This technique has two goals: it supports the user in his solution finding process with hints, but also analyzes the steps and possible

mistakes of the user. The acquired data is then used as scaffolding for adaptive guidance and presentation.

For the implementation of these features the systems has to be conceived according to the teaching subject, teaching strategies and the individual users. Therefore most AIWBES have a three-part architecture as shown in Figure 4 which is a specialization of the common architecture for adaptive systems shown in Figure 3. The domain model consists of a conceptual

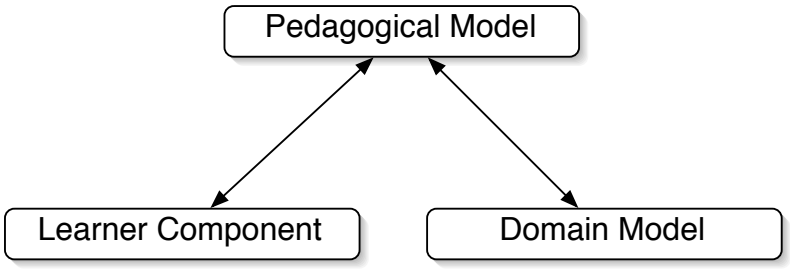

Figure 4: Typical AIWBES Architecture

network of interdependent learning units. These units are a semantic representation of knowledge of the subject to teach. For the authoring of the course material most systems provide an authoring environment to create the content. Using this approach the content is only available to the specific system and can not be reused. Because the pedagogical model, which describes the interdependencies of the learning units, is often interwoven with the learning units, a reuse is hardly possible even in the same system, e.g. for a related course. Some AIWBES already use open formats like [Soc02] which allow the reuse of learning material in another system.

The pedagogical model is only a specialization of adaptation rules described in Section 3.3, the user model corresponds to the context component.

Despite the advanced features AIWBES offer, they are used mostly in research projects and not in real life lectures. Reasons for this are the lack of some basic features most course management systems offer (communication features like announcements, scheduling features, secure user management, management features and statistics, etc.). Another reason is the work intensive creation of courses for AIWBES. The domain knowledge must be transferred into an appropriate format for the system, the interdependencies of the learning objects have to be defined and tests have to be designed. It is worth mentioning, that each of the AIWBES described in the later sections use their own format for the learning units and the learner model, with the exception of ActiveMath, which uses an extension to OpenMath called OMDoc [Koh00] for the domain model. Many of the existing AIWBES are designed from scratch for a single teaching subject, which makes them not even a second best choice for other subjects. Another flaw is the software architecture of many AIWBES, which are designed as research prototypes and not for deployment in a production environment with other requirements for scalability, security, ease of maintenance, etc.

In the following, three AIWBES are further described. Each of them has different priorities in its development, resulting in different combinations of adaptive and intelligent techniques. Because of these differences in their implementations the descriptions of these three systems

result also in a good coverage of the current status of the research and development in the area of AIWBES.

4.1.1 ActiveMath

ActiveMath [MAB+01] is a system under development at DFKI in Saarbrücken. It is a Web based learning system for adaptive mathematical courses, with the unique feature of the integration of stand-alone mathematical service systems.

ActiveMath supports the standard features of adaptive guidance and presentation, notably adaptive navigation support through link annotation and adaptive sequencing of course material with respect to the user model. ActiveMath uses the OMDoc Format [Koh00] for the representation of the domain model, an open XML format based on OpenMath and MathML. At the moment, collaboration features are not supported by ActiveMath at all.

But in spite of the lack of some of the basic features in today's prototype, ActiveMath offers some advanced features that are not found in most other AIWBES. An example is the support for adaptive choice of content. In other AIWBES, the user cannot select the topics he wants to learn, there is only one goal for every user of the system. The user can only select or self-test his previous knowledge, which defines where he enters the course. In ActiveMath the user can not only define where he wants to enter the course, but also where to leave it. The user can select what subjects he wants to learn. This makes ActiveMath especially suitable for courses in heterogeneous environments like universities, where the same subject might be taught in different courses, but with different contexts in mind, resulting in slightly different learning goals. The component responsible to this feature is ActiveMath's course generator, which is responsible to choosing and arranging the content that is needed to achieve the selected goals.

Another exceptional feature of ActiveMath is its support for the use of external services like Computer Algebra Systems. This allows ActiveMath to support interactive problem solving on a very high level without the need to reinvent existing stand-alone systems. These external services are fully integrated in the adaptive learning environment, meaning that the access to these external systems can be restricted in the exercises and otherwise adapted according to the users knowledge and the learning goal of each exercise.

Both of the above features build upon pedagogical rules, which contain know-how about the content, how to adapt the content to the user and under which conditions which external service should be accessible up to which level. For the evaluation of these pedagogical rules ActiveMath uses an expert system shell named JESS [JES02].

The architecture of ActiveMath is an open one, which enables the integration of other components developed by other projects and also the exchange of core components in a modular way. XML-RPC [Sof02] is used for the communication with external services.

Currently there is a project under development named MMiSS, which tries to integrate the assessment system WebAssign with ActiveMath. The project MMiSS is described in Section 4.4.1.

4.1.2 ELM-ART

The LISP course ELM-ART [WB01] is one of the best know AIWBES. It is based on the ITS ELM-PE (developed in the first half of the nineties) which was used for many years for LISP tutoring at the University of Trier. ELM-ART was developed to overcome the shortcomings of ELM-PE due to its platform dependent user interface, large application size and its requirements for powerful computers. Since 1999 the third generation of ELM-ART exists. This version introduced a multi-layered overlay model for the user model and some new communication tools.

ELM-ART uses a textbook metaphor. This has the advantage, that users feel familiar with its features from the beginning. ELM-ART implements all the traditional text-book features:

- hierarchically structured content
- detailed table of contents
- sections filled with textual presentations, figures, programming examples
- questions and programming exercises at the end of each section
- a glossary with brief explanations of all programming constructs
- ability to annotate sections

In addition to these features of good printed textbooks, ELM-ART adds two new features, interactivity and intelligence.

Interactivity Interactivity in ELM-ART is provided through live examples. Live examples are hyper-linked Lisp expression, following the link evaluates the expression in the interactive LISP evaluator. The user can interactively modify the expression in the evaluator for further exploration. Also a step by step evaluation of the example is possible. Answers to the questions and programming exercises at the end of each section are automatically evaluated and the user gets immediate feedback. ELM-ART can also give a detailed diagnosis of incorrect solutions, but has no problem solving support in the form of a step-by-step guidance. The step-by-step guidance would be useful for more complex problems, where the diagnosis of incorrect solutions fails.

Intelligence Intelligence in ELM-ART is provided through several kinds of support usually provided by a human teacher. Examples are the intelligent solution analysis and the diagnosis of incorrect solutions. Another intelligent feature is the building of the optimal learning path according to the knowledge of the specific user. ELM-ART uses adaptive navigation support through link annotation to suggest the user the next section to explore. The user model used to adapt to is built upon the solution provided to questions and exercises and upon the sections already accessed by the user. Due to the overlay model used in the user model, the user is able to change his knowledge level without altering his test results.

Computer mediated communication supported by ELM-ART includes a chat and a discussion forum. More advanced communication forms that allow adaptive collaboration and awareness of other users working in the same section are not implemented in ELM-ART.

Authoring Systems like NetCoach [WKW01] or ART-Web allow authors to develop adaptive learning courses based on the technique of ELM-ART without programming skills. ART-Web was used to develop RR2000 [RR201], a course for the new German spelling rules, and Wein-ART [WA99], a wine information course. An additive feature of NetCoach over ELM-ART is its support for Tutors managing courses through a Web-interface. Tutors can also observe users in the course. They have access to the current learning state of the user. That is how long each user has been working on the course, which messages he sent to the tutors, which concepts each user has worked on, how many errors he has made and some other informations. This information could be very helpful to the tutor in analyzing the students' success, but only this raw data is accessible and there are no means for a statistical analysis.

4.1.3 WebDL

WebDL is developed at UNED, Spain's distance university. WebDL has a completely different architecture than the other two AIWBES presented before [BG00]. It is based on a multi-agent architecture, or more specifically, a multi-agent decision system. The system architecture consists of two main components, a user interaction component and an adaptive module. The user interaction component consists of an interface agent which organizes the different contents that should be part of the response. The adaptive module consists of three groups of agents:

- One group provides the basic functionality of an AIWBES. It consists of a user model agent (corresponds to the user model of the AIWBES architecture), a user modeling agent (also corresponds to the user model), a material agent (corresponds to the domain model) and a pedagogical agent (corresponds to the pedagogical model).
- The next group consists of agents for identifying which system services are of interest to the user. This is done by collaborative filtering of the users' behavior.
- The last group consists only of the coordinator agent, which is responsible for distributing the user request to the different agents and also to collect their responses and for forwarding them to the interface agent of the interaction component.

Besides the standard features of AIWBES implemented in WebDL like curriculum sequencing, adaptive presentation, adaptive navigation support, the feature of adaptive collaboration is exceptional. It allows to put students of similar interests and knowledge levels in touch with each other, e.g. by setting up work-groups for them. There exists also an integration of WebDL with a collaborative Web application called aLF [BGC01] that is build with the content and collaboration software ArsDigita. This combines the adaptive educational features of WebDL with the features of community systems including work-groups, newsgroups, chats, bulletin boards, calendaring, project management and document management. Some of these community features like work-groups and chats can profit from the adaptive collaboration features of WebDL

4.2 Course Management Systems

Course Management Systems (CMS) offer services for the management of Web and Web-enriched courses. They are used by many universities, especially in the US (Carnegie Mellon,

MIT, Stanford, etc.). They offer the teaching staff the ability to manage their courses on-line. The common features of course management software are:

Student Management: Students can be required to register for courses. This allows the tracking of student data like course attendance or students' performance in exercises and tests to be recorded automatically in a students' grade book.

Content Authoring and Management: CMS offer the teaching staff the possibility to publish teaching material for courses to the Web. Existing content in formats like PDF or HTML can be added to a course's material section to make them accessible for the registered students.

New content can be created using an authoring environment and existing content can be converted to be used with the CMS. The authoring environment includes features like a course structure editor, a curriculum manager to define the learning objectives and authoring support like index creation and the support for multiple authors.

The content managed within the CMS is keyword search-able by the students.

Exercises: Exercises are supported by CMS in two forms: small quizzes and human evaluated more complex exercises. Quizzes consist typically of simple multiple-choice tests, fill in-the-blank tests, or other short answer tests. The quizzes are authored with the authoring environment. More complex exercises like programming problems can be submitted for human evaluation.

The results of the exercises can be manually or automatically transfered in the students' grade book.

Progress Tracking: As all student actions are tracked by the system and student data like grades can be managed with the system, CMS are able to allow individual students to track their progress and performance in a course on-line.

Communication and Collaboration Tools: CMS support various forms of communication, both student to student and student to teacher. Typically email, discussion forums and chats are integrated in CMS. Collaborative learning is supported through shared workspaces.

As stated in [BM01] and [WB01], most of these typical features of CMS are provided in a more sophisticated form by AIWBES. The question that arises, is why CMS are used by many institutions in many courses, and not AIWBES. One answer is the longer availability of CMS. Versatile AIWBES, which are suitable for practical Web-based education were not available at all before 2000, and also today's versatile AIWBES like ActiveMath and ELM-ART lack some features that are important for a widespread adaptation:

- support in form of end user documentation, technical support, workshops and other services
- easy to use by not technically educated teaching staff after an initial setup of the system in the institution by an administrator
- privacy and security support

- support for emerging standards from organizations like IMS

In the following sections first two commercial CMS are presented (WebCT and Blackboard). After that an academic implementation of a CMS is presented (AulaNet), which is especially interesting because of its groupware approach. Then two open source projects backed by universities are presented (OKI and CampusSource).

4.2.1 WebCT and Blackboard

WebCT and Blackboard are the two commercial implementations of CMS that are most widely used. Both are very similar, they have all the typical features of CMS listed in the previous introductory section. But the implementations of some features differ:

Content Authoring and Management In the area of content management, the two systems have the greatest differences. Blackboard's content management for course material consists only of a hierarchical repository for up- and down-loadable resources of any format. WebCT offers, in addition to a repository similar to Blackboard's, an integration of HTML content in the system. This enables WebCT to offer exercises (in the form of quizzes) integrated in the course material and communication features like discussion forums linked to specific topics. Another advantage of WebCT is the possibility of a time or student performance driven publication of course material, while Blackboard needs the instructor to release material manually.

Communication and Collaboration Tools As already mentioned, Blackboard offers communication tools attached to the course material. Other, non-content related communication tools are offered by both systems in a similar way with one notable exception: WebCT has no real support for e-mail. It offers only a system internal e-mail functionality. Blackboard instead needs external e-mail addresses of the users and has the possibility to send e-mail to these external addresses through the system.

Real collaboration tools are only offered by Blackboard. With Blackboard it is possible to set up sub groups of students with sharing and communication features which enables active collaboration of the students through the system.

Both WebCT and Blackboard support and promise to support the standards for learning systems developed by the IMS Project. Blackboard is one of the founding members of the IMS project. Thus, WebCT is able to import exercises in the format specified by IMS Question and Test Interoperability. In addition Blackboard announced to support the API's developed by the OKI project to reach interoperability and allowing applications developed on top of the OKI API's to run with Blackboard.

4.2.2 AulaNet

AulaNet is a system developed at the Catholic University of Rio de Janeiro. The current version of AulaNet (2.0) is distributed commercially while a previous version (1.3) is available for free. AulaNet differs from other CMS because it is based upon a groupware approach and tries not to rebuild the traditional physical metaphors like blackboards and classrooms.

As a groupware system for electronic education it is based on communication, coordination and cooperation. The communication features of AulaNet include the common synchronous and asynchronous communication forms like e-mail, discussion forums and chats. Coordinating groups is supported by scheduling tools that allow the announcement of events such as a chat or a deadline. Another tool to support the coordination of groups are assessments that can verify the progress of a project according to the schedule. Cooperation services include co-authoring support for learners and teachers. Teachers are enabled to cooperate in the creation of learning content with other teachers or learners, learners are enabled to cooperate in the creation of additional learning information.

The current version of AulaNet, Version 2.0, is implemented using the Java servlet technology in conjunction with Scriba (an HTML embedded scripting language for accessing databases, similar to, but not as powerful as JSP). The usage of the embedded scripting language and Java-Script in the HTML pages in combination with the steadily growing list of features led to a no longer easy maintainable and extend-able software system. Currently, a successor to AulaNet, eLabora, with a component based architecture is under development. It shall be based on a component framework with basic component coordination services for persistence, transaction, etc. On top of this component framework component type specific contracts and interfaces are defined. These interfaces are implemented by the components. The eLabora project team hopes that by the development of such a highly modularized, extend-able and adaptable software system other institutions that use the eLabora system are able to develop new components that can be coupled with the existing system.

4.2.3 OKI

OKI [oT02b] is a development project led by the MIT in collaboration with Stanford and other institutions in the US and in the UK. The initiators of the OKI project recognized, that existing commercial and academic platforms for learning management are not very extensible, in respect to new educational application as well as in respect to the integration in university's back-end infrastructure. The primary goal of the OKI project is to develop an architectural specification building a framework and foundation for educational applications as open source, resulting in an open and extensible standard for learning management platforms. In order to reach this goal, the project's first step is to define two API layers, the common services layer for integration with existing back-end infrastructure and an educational services API on top of the common services API as the base for educational applications. The project hopes, that other actors in the educational community could be attracted to contribute tools, services and educational applications using this framework or conforming to the defined API's. To bootstrap the contributions, MIT and Stanford will port their existing course management systems (Stellar and CourseWork respectively) to OKI in order to create an initial implementation of the framework and also a set of exemplar applications.

To reach a broad adoption, OKI supports existing or proposed standards and specifications and collaborates with different standards bodies. OKI is e.g. a member of the IMS Project and supports its efforts and will adopt their standards. The open and layered architecture of OKI, described in more detail in Section 6.2.1, with its integration into existing back-end systems and its provided common services like user profile management, content management, etc. shall prevent the need to re-implement common functionalities again and again in every new system. Contributors are instead enabled to concentrate on the development of new

educational applications.

The OKI project with its strong backing will surely play a major role in the area of course management systems. But it is questionable, whether to define an infrastructure for educational applications through an API in a specific language, i.e. Java, is the best way to go. The interoperability and extensibility would benefit from language independence. Another weak point is, that major resources are needed to rewrite existing applications to make them conforming to the OKI API's and usable together with the OKI system. A more open approach of the OKI infrastructure with respect to how it can be used would allow to reuse existing applications with only minor changes or through a small compatibility layer.

4.2.4 CampusSource

CampusSource is not a single project, but an initiative to support the development of open source course management systems. The goal of the project in the long run is to integrate the different independent projects to get a complex software system, that supports the needs for a so-called virtual university.

At the moment eight different projects are hosted by CampusSource. Their functionality reaches from assessment systems, homework submission systems to data warehouse systems for universities. The following lists some of the projects:

WebAssign WebAssign is an assessment system. It is described further in Section 4.3.1.

OpenUSS OpenUSS [Ope02] stands for Open University Support System. It is a Web-portal for faculties and universities. The portal offers communication and information services to the students via various communication devices like Internet PCs and WAP phones. These communication and information features are personalized: students are able to subscribe to different information sources like lectures.

The organizers of lectures are supported in publishing documents like lecture notes or exercises and in announcing lecture specific events. Students can be automatically informed about new documents available.

The system is designed according to the application service provider (ASP) model, that means that one installation of the system can handle more than one lecture or faculty. Organizations that have no resources to run OpenUSS at their own can out-source the system to a different entity and use one instance of OpenUSS together with other organizations.

The architecture of OpenUSS is a three tier architecture based on Sun's J2EE architecture [Mic02b]. It uses Enhydra [Obj02a] for presentation purposes, JOnAS [Obj02b] or jBoss [Gro02b] as application server and embedable databases like Interbase [Cor02] or HypersonicSQL [Mül02] for data storage and retrieval.

ILIAS ILIAS (Integriertes Lern-, Informations- und Arbeitskooperations-System) [ILI02] is a system for the development and the usage of Web-based educational courses. It implements the content authoring and management functionalities of CMS and offers the communication and collaboration tools typically found in CMS systems. Its features in this areas are at least on par with those found in commercial systems like WebCT or Blackboard. ILIAS supports the meta-data descriptions defined by the IMS Project (see

Section 3.2.1), team authoring, automatic publishing of news and changes to learners, offline versions of course material, etc.

ILIAS is implemented as a PHP Web-application with MySQL as database.

There exists a high level architecture of an integrated system for the virtual university. Unlike OKI, CampusSource architecture builds not on a common services layer, but consists of mainly self-contained and independent applications. These applications make use of existing infrastructure like directory services and databases. They communicate with each other via a programming language independent protocol, i.e. CORBA [Gro02c], to benefit from and make use of the services available. To get a common user interface to this independent but cooperative applications, an integration application exists which is just another independent cooperative application but with the difference, that it not only communicates to the other applications via CORBA, but also integrates the Web interfaces of the other applications to a common one presented to the user. This approach is a more flexible coupling of different applications than the tight coupling via API's as done by OKI. The weak point of CampusSource architecture for the virtual university is, that up today only a high level description of the architecture exists, no implementation or a more in-depth feasibility study. The CampusSource architecture is discussed in more detail in Section 6.2.2.

4.3 Assessment Systems

Assessment systems are in contrast to course management systems very specialized applications with only one field of activity: assessments. But for this specific field they offer often a range of features not found in other more general systems. Assessment systems are typically developed at universities for the actual needs in lectures and appended tutoring sessions. They replace the traditional methods of paper based homework and manual paper based corrections by a system of electronic homework submission, semi-automatic corrections and tests and also support for manual corrections and administrative tasks.

An example for such a system is the one developed by the author of this diploma thesis as part of his project work [Sch01]. Two other systems are presented here: WebAssign and Praktomat. WebAssign is a typical assessment system and shall become part of MMiSS (Section 4.4.1), too. Praktomat has a smaller area of application, but also some interesting features which justify a more detailed description.

4.3.1 WebAssign

WebAssign is a system developed and used by the Distance University of Hagen. It is one of the projects supported by CampusSource (Section 4.2.4). It supports all steps in conjunction with exercises for tutorials:

creation of exercises The system supports three different types of exercises: multiple choice, free text and programming exercises. The multiple choice exercises are corrected and graded automatically by the system, the programming exercises can be pre-checked by the system at submission time and the free text exercises need to be corrected and graded without support of the system. These exercises need to be created with the

Web-based exercise creation assistant of WebAssign. It needs a PUT request capable browser like Netscape's Composer. An import or use of external exercises is not possible.

download of exercises by students Each exercise has an associated time that specifies when the exercise should be available to the students. When this time is reached, the students can download the exercises or answer the questions only, depending on the exercise type.

submission of solutions to the system Each solution to an exercise, edited on- or off-line by the student needs to be explicitly submitted by the students. For on-line edited solutions the student only needs to press a submit button, off-line edited solutions needs to be uploaded by the students with the help of an HTML form. Each exercise has a time specified until which the students are allowed to send the solutions back to the system.

distribution of solutions to the correctors After a student submitted his solutions, they are immediately delivered to a corrector. There are different distribution strategies available. All of them take as primary goal the work-time of the correctors into account. As an additional secondary goal it can be specified, that a student should get the same corrector for every part of his solutions to an exercise, or for every solution at all.

correction by the correctors After the distribution of the solutions, the correctors can access them through WebAssign. WebAssign allows two methods of correction: on-line correction in a PUT request capable browser, or off-line correction with the help of a locally installed server. The corrector grades the exercises, adds additional comments and then inform the system that he is finished. The graded solutions are then sent back to the system.

information of the students about the result After the solutions are graded, the students get informed about their grades by e-mail. The students can access their prior grades through the WebAssign application.

storage of the solutions and grades in a database The exercises, student solutions and grades are stored in a database for archiving purposes.

WebAssign is a closed system in respect to the usage of external tools for the creation and correction of exercises. All administrative work needs to be done through the Web-interface of WebAssign. This is comfortable for first time users, but not efficient for regular users as administrators are. A method to use specialized external tools for the creation and correction of exercises, for the configuration of the application and for evaluation of the grades would be preferable to such closed all-in-one approach.

WebAssign is implemented as Java Servlet [Mic02d] application. It offers two CORBA interfaces for the communication with other applications. One CORBA interface allows other applications to register new courses within WebAssign which includes the registration of students for this course. The other CORBA interface allows the use of external automatic correction services. These services are called at solution submission time by WebAssign and get the solutions of the students and respond to WebAssign with the corrected and graded solutions. There also exists a interface to LDAP [YHK95] services and HIS (Hochschul-Informations-System) [Gmb02] for the authentication and authorization of the students.

WebAssign shall become part of MMiSS (Section 4.4.1).

4.3.2 Praktomat

Praktomat was developed at the University of Passau, Germany. It is not used for regular lectures but instead for a programming course. One consequence is that the system only supports homework with solutions consisting of the source code of computer programs. Another one is, that the students have about two or three weeks for their homework, not only one week as in more typical courses.

These restrictions and course specific properties are used by Praktomat to offer some advanced features:

parameterized tests Praktomat offers to parameterize tests. This is done with macros. This feature is a prerequisite of the peer review feature described below. But also independent of a peer review feature this is useful for the prevention of plagiarism.

automated testing Praktomat allows automated tests of the solutions. It distinguishes between public and private tests. Public tests are performed at solution submission time. If the test fails, the student gets a second chance to send in another solution. The private tests are performed after the students have submitted their solutions. The results are only available for the correctors.

peer review The most interesting feature of Praktomat is peer reviewing of student solutions by the students themselves. After students sent in their own solution, they get access to a solution of another student (with other test parameters to avoid plagiarism). The students are then asked to review the other student's solution. If a student has finished reviewing a solution and added some comments, the student gets an additional time to reevaluate his own solution.

All these features need a longer period of time between problem formulation and the submission of the solutions. The tutors need some time for the comparatively complex and time intensive task of the creation of parameterized exercises and for writing the automated tests. For the peer review to be successful, the students need some additional time between the submission of the solutions and the new problem formulation.

For the creation of the exercises and also for the creation of the test cases Praktomat offers a Web-interface. At first sight this looks like a good feature, but a Web-interface is not designed primarily for data input and text edit. For this task there exist very sophisticated applications. Making the exercise creation and the test case creation possible with external tools would be a real feature users benefit from.

In comparison with other systems Praktomat lacks some features. A small course management part, that allows the storage and management of the results of the individual students would be a very useful feature which is present in other assessment systems. Also the system integrates the problem formulation and the submission of the solutions very tightly. A separation of information needed by the student and information only needed by the system would help in the creation of the exercises. It would allow to reuse exercises independently developed with Praktomat by enriching them with additional information needed by Praktomat.

After the first use of the Praktomat system in a programming course, a survey on the acceptance of the system by the students was made. The feedback was very positive, especially the peer review feature of Praktomat was often mentioned as a very meaningful and useful one.

4.4 Integrative Systems

The systems described in the previous sections all cover different aspects that would be useful in the area of teaching services. The AIWBES offer great tutoring functionality but lack useful management functionality, the CMS concentrate on management functionality but offer only basic support for tutoring and also have no adaptive functionality, and the assessment systems concentrate on a single aspect.

The question is, why not to combine existing systems of each category in order to create a system that covers more of the useful aspects than any of the existing system does. There exists one project with the goal to combine two systems of the AIWBES and the CMS area: MMiSS.

4.4.1 MMiSS

MMiSS is a project with the goal to combine the AIWBES ActiveMath (Section 4.1.1) with the assessment system WebAssign. It is a collaborative project of some German Universities (Universität Bremen, Universität Freiburg, Fernuniversität Hagen, Ludwig-Maximilians-Universität München, Universität des Saarlandes). MMiSS stands for MultiMedia-Instruktion in Sicheren Systemen. This shows that the primary project goal is not a versatile system that is usable with different teaching subjects, but a system specific to one subject, similar to the AIWBES already available. In accordance with this premise and in the tradition of the AIWBES, the project also includes the creation of new course material for the use with the system. For this course material a new proprietary format is currently being specified. The disadvantages of a own document format are described more deeply in Section 3.2.1.

At the moment the project is at the very beginning, but initial architecture proposals build upon the architecture of ActiveMath which already combined different individual applications through XML-RPC communication. This is a very promising approach, because it not only needs no common programming language like OKI's approach, but instead uses a text based communication which can be used over existing Internet protocols. This not only results in a great programming language independence but also allows the use of existing infrastructure and also security features for the communication.

The project has the potential to form a basis for further development and integration of educational tools forming a versatile and adaptable software system which can be used in many areas. The prerequisite for such a success is, that the project will be modular enough to replace the content component which currently seems to be tightly coupled to a specific teaching subject with a more versatile one that uses a meta data description of the content instead of defining its own proprietary and subject specific format.

5 A Tentative Classification of Teaching Services

The enumeration of all the features provided by the systems described in Chapter 4 would result in a huge list. But in addition to these already implemented features and services, there are also new ones which are until today not implemented in systems that support teaching and learning. Most of these features are not only features which are eligible for systems in the educational area, but also in many other areas. In this chapter some of those services are presented and classified. But at first the criterias for such a classification are discussed.

5.1 Usage Centric Classification vs. Service Centric Classification

The range of services that are useful in the domain of teaching includes services supporting the faculty staff. Other services bring surplus value to the students and some of them bring benefits to both groups. To classify these services according to these two user groups, perhaps with the addition of system support staff as third group, is quite obvious. And indeed such a classification of services is found in comparisons of course management software [Lan02], [Sie02], [Edu02].

But this obvious classification is limited. In many tasks, all of the groups work together, but certainly from different directions. An example would be a service that lets the student see his performance in comparison to other students over time as an indicator for the effectiveness of his learning. The same information, presented in another way with cumulated data, is very interesting for the teaching staff. With this in mind, in the following sections a classification by functionalities rather than user groups is used.

As already noted, most of the services that are useful in the area of teaching services are not specific to teaching. For example most of them are relevant for cooperative work in general, with small but worth to mention differences: the roles and the associated rights of the participants. There is a clear distinction between the roles of the students and the staff, but in most cooperative work scenarios there is no such clear distinction, there is only one team, which is perhaps organized by some hierarchy.

5.2 Teaching Material

Every lecture uses some teaching material. It includes lecture notes, exercises and links to external resources. For the management of this material various services are imaginable. The basic requirements are the storage and the delivery of the teaching materials to the students.

5.2.1 Content Storage

To be accessible by the students and to be available for other services the content needs to be stored in the system. Requirements for the storage include the support for various data formats, the possibility to enhance the content with meta data for the use with other services and common storage requirements like reliability and security.

Section 3.2 describes various techniques that can be used for the storage of content.

5.2.2 Content Presentation

The stored content needs to be accessible to the students. Some content may be only accessible for a specified amount of time and some only to a specified group of students.

As the system stores the content it is also aware of updates to the content. Students can be informed about the updates. If the system knows about the kind of update (e.g. spelling mistakes or content mistakes) it can decide if and in which way an information of the students about the update is necessary.

Additional services of content presentation are possible if the content is enriched with meta-data. Meta-data enriched content can be presented in various forms depending on the users' needs and contexts. Some of the possibilities are described in the section about adaptive hypermedia (Section 2.3).

5.3 Individual and Group Browsing

Browsing services are mainly based on analyzing the navigation behavior of the users. With the collected information it is possible to support the users of the site in navigating the site. On the other hand the authors of the navigated content benefit from the feedback they get by analyzing how their content is used. In Section 2.1.4 the technical requirements for the support of group browsing are listed.

5.3.1 Suggestion of Related Information

Based on what other users have browsed before and after they visited the actual page, it is possible to show the user what pages are related and he might find interesting, too. This information can be made more reliable with a page rating system. Such a rating system can ask the users explicitly to rate and classify the current page. But rating systems can be also implemented without the requirement of explicit user ratings: a page rating system can also be based on factors as how long a user stayed on a page, whether he visited this page already in the past or not, or whether or not he used some features offered by the system in conjunction with the page like annotations, discussion forums and bookmarking.

From this information about useful related pages, not only the students benefit, but also the teaching staff: If, e.g. many students reading a chapter of the lecture notes search for additional information elsewhere, perhaps the notes should be improved in this area.

That such a service is not only relevant for teaching or the broader context of cooperative work shows the integration of similar services in e-commerce sites like Amazon. On Amazon you get recommendations of books based upon what other customers bought ("Customers who bought this book also bought ...") and upon what other customers where interested in who also looked at this book ("Customers who shopped for this item also shopped for these items ..."). The first of these two examples is based on explicit user rating: a user who buys a book is surely interested in the book, while the second example is based on assumptions of the system.

5.3.2 History Functionality

All browsers have some history functionality, but this functionality is very limited. The user only gets a list of pages that he visited in the past, sorted by domains and/or time. The user has to dig through a huge amount of information to find useful information.

A more sophisticated history functionality would be one that is not only based upon pages the user visited, but also based on the subject of these pages. By grouping the visited pages by subject, number of visits and time of visits a kind of personal start page could be built. E.g. to a student reading through lecture notes, going from on page to another, the system could present him on his start page, perhaps on his next login some days after the last access, the link to the last chapter he read. A system aware of the user's past action can also inform the user about changes and news that are linked with these actions. If there were updates to some chapters the student already read, the student would be delighted to be informed about, but if he didn't access the updated chapters in the past he probably is not interested at all.

An advanced history functionality could evolve into a recommendation service for relevant content based on the user's past actions. Again the example of the student reading through the lecture notes: if a new chapter or addition to an existing chapter is published, he probably is interested. This can be combined with a service that ensures that all users are informed about changes and updates that are relevant for them. Another possibility is, that the teaching staff explicitly recommends the reading of some pages, e.g. until the next teaching session. To students that have not yet read these pages links to these pages are presented.

That such advanced history functionality is also relevant in the area of e-commerce systems is shown by another example of Amazon. Amazon presents the users the last viewed products and categories in the "Your Recent History" section at the bottom of each page. They also create some kind of personal page with the user's history and a suggestion of related products.

5.3.3 Private Annotations

The ability to make private annotations (for public annotations see Section 5.4.2) is a great benefit for frequent users of a page. Most people make some personal comments or do some highlighting while studying a text. A service that allows such personal annotations also for on-line reading would make downloading and printing of the text solely for this purpose no longer necessary. Downloading and printing of the text has besides the additional effort other drawbacks: if the text is changing, which could be the case with lecture notes, which are often evolving during the semester, the student needs to merge his notes with the new text manually. Another drawback is, that for printed texts, the other features described in this chapter are not available. Examples are on-line texts that are enriched with asynchronous communication features.

5.4 Communication

Communication services can be categorized whether the communication is synchronous or asynchronous. For a successful synchronous communication service a great user base is needed so that at least two users are interested at a time on the same subject. Because this is not likely

in the domain of teaching services, where the potential user base is equivalent to the number of students plus teaching staff, in the following section only different forms of asynchronous communication services are described.

5.4.1 Discussion Forum

In a discussion forum only the topic is predefined, the specific subjects that are discussed are defined by the members of the forum. Forms of Web based discussion forums are Usenet groups and mailing lists. The Usenet postings are preserved on the Usenet server for some period of time, the posts in mailing lists are only received and stored by the members. So after some time all messages are no longer accessible, unless the users themselves have preserved them. An implementation of a discussion forum as a teaching service should preserve the messages and make them easily search-able and browse-able.

In providing a Web based front end to the discussion forums based upon mailing lists or Usenet groups, these forums could be made accessible for all (including the new) users, while preserving the accessibility of the forums with better-suited tools for the regular users. An example for such a Web based front end for public Usenet groups is Google's group section [Goo02a]. Google's front end to Usenet groups allows to search for and read through past messages and also to post new ones. There also exist archives for public mailing lists ([Bre02], [Gro02d]) that allow searching and reading through past messages, but no posting because posting is commonly only allowed to subscribes of the individual mailing list.

5.4.2 Public Annotations

Annotations to existing content, e.g. lecture notes, are another form of asynchronous communication. The students are enabled to ask questions or provide additional information to specific points of the lecture notes right in place. Other students or the teaching staff can answer the questions, the teaching staff can integrate the additional information generated by the comments in the next update of the lecture notes.

For the students it is more comfortable to ask questions right in place while reading the lecture notes. In contrast to the discussion of the question at a different place, e.g the discussion forums mentioned above, it is guaranteed that other students reading the same sections notice the discussion, enabling them to provide an answer or profit from given answers.

An implementation of public annotations to books can be seen at Andamooka [And02], a site that "hosts open content books for reading, annotation, and discussion".

5.4.3 User Moderated Pages or Wikis

User moderated pages are pages, where the users themselves create and modify the content. Every user has the right to modify the pages, right in place, right in his browser. The result is a kind of discussion forum, but not in the strict question and answer form, but more like a network of informative pages that evolves and grows over time. This is the idea of the so-called "WikiWikiWeb", the first and biggest implementation is the "Portland Pattern Repository" [Cun02].

This example shows that a simple to use collaborative authoring service may result in very valuable resource. One key factor to the success of such a service is, that the boundaries between reading and authoring are very low. An intuitive way to enhance the content is needed to allow readers to switch to the author role without much effort.

5.4.4 Announcements

Announcements are a special form of communication, because they are only one way: from the teaching staff to the students. Announcements made through the system could be delivered in various ways: shown on the Web site, delivered by e-mail, posted to forums, etc. It could also be tracked how many students have seen the announcement and/or explicitly confirmed its reception. An announcement service is most valuable if it is integrated with the other teaching services. Two of them are mentioned above: the various communication channels and the access logging facility which is also needed for the browsing services. An integration of the announcement service with the content management services would allow to automate the informing of the users about new resources available. Time driven announcements are possible, if a scheduling or calendaring system has access to the announcement service.

5.5 Homework Management and Assessment

Many lectures have associated tutorials with exercises. The students work on their solutions to the exercises at home, alone or in a small group. Some of these solutions are graded, the grades received might act as requirement for further grading.

Although this seems very specific for teaching, variants of some of the services described below are applicable in the domain of team work, too. Like the solutions that have to be submitted until a given date, in a team some tasks have to be finished until a given date. Previous work can act in both cases as prerequisite to further work. Also reviews take place in team work. Those analogies (and distinctions) will be mentioned in the further discussion.

5.5.1 On-Line Accessible Exercises

The exercises have to be made available to the students. This can be automated by defining a schedule for the availability of exercises. The exercises can then be accessible to the students by the Web or sent to them e.g. by e-mail. If there are errata, additional comments or something similar, every student that already accessed the exercise could be automatically informed.

Another surplus value of the on-line accessible exercises is, that they can be parameterized to prevent plagiarism. This also enables student peer reviews as discussed below.

5.5.2 Submission

An on-line submission is required to automate the further processing of the solutions to exercises as described below. But also on-line submission in itself generates surplus value for both the students and the teaching staff.

The students gain location independence. They do not need to drop their solutions in a mail box located at the university, they can simply submit them from the place they where done: e.g. the computer at home. This means for the organization of the courses by the teaching staff, that the deadline for each solution could be chosen regardless of the presence of the students (and the teaching staff) at the university.

Another benefit for the students is, that more than one solution can be submitted. New solutions can automatically override older ones. If there is more than one solution to submit until a given date, the students can also make incremental submissions of their solutions. These enhancements above the offline submission of solutions are fully transparent for the teaching staff. They only get one solution for each exercise from each student.

5.5.3 Peer Review

Peer review in the context of student solutions means, that the students review each other's solution to the exercises. There are many benefits for both the reviewed and the reviewing student, especially with programming courses:

- The students learn to read and understand code, not only to write. This is an important learning goal of programming courses.
- The students learn to review code. This not only includes understanding code, but also spotting problems with the code and articulating the problems to the original author. This competency is needed by every programmer.

  This benefit is not special to programming courses and coding. Reviewing other people's work is a competency needed in most other activities, too.
- The students get aware of the need for writing clean code. This includes the accordance to coding guidelines, the use of meaningful variable, method and function names, and code documentation. Students, who attempt to understand code of other students will get convinced by the benefits of clean code very quickly.
- Finally, the correctors get unburdened.

Such a peer review mechanism is already successfully implemented for programming courses in a system called Praktomat, discussed in Section 4.3.2.

Peer reviewing is applicable to many domains, but especially in software development, regular code reviews are implemented in most process models as method to improve the code quality and also to become not too dependent on single programmers. Especially applicable are such peer reviews for security relevant software products. Open Source software has the ambition to be more secure than closed source software, because the source is accessible to and reviewed by many people. This should help to discover and correct design flaws and bugs more quickly. An example for peer reviewing in another domain are a number of scientific journals that want to benefit from "the free access" that "guarantees a large audience, while keeping the high standard of scientific publications" [DMT02]. Interesting in this context is also the development in the area of cryptography. While encryption was based until the second half of the 20th century almost exclusively on secret algorithms, today's encryption algorithms can

be peer reviewed and checked for errors by everyone. Instead of relying on obfuscation and on secrets, their reliability can be verified with mathematical procedures.

5.5.4 Automated Testing

The automated testing of the student's solutions can be implemented in two stages: when the students submit the solutions to get instant feedback, and second before the solutions are distributed to the correctors.

The students benefit from instant feedback they get for their work, while the correctors get test results for each solution. In programming courses the tests at submission time may ensure that only syntactically correct code that passes some basic tests can be submitted. This can be compared to a software project team and the submission to the source code repository. To be allowed to check in new or updated code to the repository, there is usually the requirement that it passes all tests of a predefined test suite.

There is another sort of testing that is applicable to students solutions beyond the above functional tests: plagiarism detection. Depending on the kind of the solutions different forms of plagiarism detection are possible. More advanced plagiarism detection services like JPlag [PMP01] for programming source code offer good results only for larger programming exercises. But also plagiarism detection in the form of a simple textual comparison is informative for the teaching staff and may help to prevent the most obvious forms of plagiarism.

5.5.5 Correction and Grading

Finally the solutions of the students have to be corrected and graded. If the corrected and graded solutions get back into the system, they can be automatically delivered to the students and the actual grades can be stored for later processing, e.g. for organizational tasks like the permissions to take part in exams or for statistics that are described in the next section.

5.6 Progress and End-of-Course Statistics

Every on-line interaction with the system creates data that can form the basis for various statistics. These statistics can help to improve the system by analyzing how the users actually use it. But also the users themselves can immediately benefit from various statistics described in the following sections.

5.6.1 Progress Tracking

By analyzing when the students submit their solutions of the exercises in correlation with their grades over time, information could be gained that allows inference on the learning progress of the students. While this information is interesting for the teaching staff, the student might be interested in his own performance in comparison with the average student. An alert functionality for the student could be implemented, which informs him, when his personal performance drift from the average student.

Such a progress tracking service helps the student to improve his study skills. It can also function as an early indicator to teaching staff for problems with specific subjects.

5.6.2 End of Course Statistics

At the end of the course, there is more data available for analysis than during the course. In addition to the data acquired by the system other information like the performance of the students or the written examination at the end of the course may be available.

With this additional data a more in-depth analysis of the success of the course is possible. The drawback of such an end of course statistics is, that the result of the analysis can not be used to improve the course itself. Only subsequent courses can profit from the results.

5.6.3 Usage Statistic

The teaching staff can get additional information how well suited the content is they provide by analyzing how it is used. One of the questions is, whether there is a correlation between the exercises the student works on and the part of the lecture notes accessed or not. The answer to this question would help to improve the provided material. Deficiencies of the teaching content could be spotted earlier and removed during the actual semester.

5.7 Course Management

Course management services simplify and support the administrative tasks that occur when offering a course. Many of the services already described above can be seen as course management services, too. The management of course material and the information of students about changes to the course material or the support for the delivery of the course material to the students are two examples of tasks that can be described as administrative, too.

In this section purely administrative tasks are described which therefore do not fit in any of the other categories.

5.7.1 Registration

Most of the services need to know who accesses them, therefore a user (student) registration system is needed. Most likely there exists already a database of the students for other purposes. A course registration service can make use of this existing database and avoid duplicate work and data inconsistency.

Not only other services need to know about the registered students, but also for the management of a course a list of all students is needed in many cases.

5.7.2 Admission

For some courses the students may need some prerequisites to be allowed to participate. For the successful completion of the course the students may need to participate in a final examination. And to be allowed to take part in such a final examination the students may need

to reach a specific number of points in exercises or a specific number of entries in attendance lists.

This shows that many different factors before and during a course may be prerequisites for the participation. An admission service therefore needs to be very configurable and also extensible enough to allow the introduction of new prerequisites. But also manual intervention should be possible, because there might be some cases that are too special to configure through a rule. An example is the case of a blind student, who can not take part in an ordinary examination. Such a service can automatically inform the students about the status of their admission and might be used for the generation of various documents needed for organizational purposes.

5.7.3 Document Generation

Various documents need to be generated for the management of a course. Examples are attendance lists, exercise result lists and name tags for the seating during an examination. At the end of the course the successful students get a proof of attendance.

The generation of such documents needs the student registration and the admission service already mentioned.

6 Towards an Open Architecture for Composable Teaching Services

The previous chapters have shown that in the area of teaching services different research topics are relevant, different components are needed, many systems have been developed, and that many different services exist. The question is, how these different aspects can be combined to a powerful, in the sense of being as useful as possible, solution in this area. An answer to this question will be developed step by step in the following. At first the question will be formulated in more detail: goals will be presented that an architecture for teaching services should fulfill to be successful. After this listing of the goals with rationales for the goals in the second part, two existing architectures for teaching services, OKI and CampusSource VU, will be checked against these goals. A resume about the results of these checks will close the second part of this chapter. The third part will show how an architecture that could fulfill the goals presented in the first part could look like. The architecture presented is a proposal of an open architecture for composable teaching services. It is an architecture that does not ignore the existing work in this area, but instead tries to combine and integrate existing solutions. The architecture presented below in Section 6.3 does not ignore and will not supersede existing systems. Instead it wants to integrate them. Possibly, current and future developments in this area would be influenced by this architecture.

6.1 Goals of an Open Architecture for Composable Teaching Services

The introducing words to this chapter already mentioned that many different aspects exist in the area of teaching services. Some of them need to be considered and also should be considered in an architecture for teaching services. To allow the consideration of these different aspects, the primary goal of an architecture that wants to be successful in this area should be, in the opinion of the author, to be open and composable. What means "open and composable"? It means to be able to integrate and combine existing systems, to be able to make use of existing resources like teaching material, to be open for further enhancements, to have no one-size-fits-all attitude, but instead to allow the adjustment of the systems to the individual needs by combining individual components. The individual components should be seen as independent services that can be combined and integrated by the open architecture to a new system. The services approach allows the individual components not only to work together with the combining and integrating system. It will also allow the components to use each other directly as services without any indirection. The services approach also allows to use the components by the end users not only through the integrating system, but also individually or in combination with other applications.

Chapter 4 presented different systems. Although the systems were selected to be representative for similar systems and although most of the presented systems have at least one single feature not found in other systems, many features are found and implemented in the presented systems in a very similar way. This shows how much duplicate work has been done in this area in the past. To do this work once again should not be the goal of a new project. An open and composable architecture will profit from the existing work already done in this area and thus it will avoid duplicate work. This must be the superior goal of a new architecture.

The following lists the goals an architecture needs to fulfill and also gives rationales why the

fulfillment of these goals is essential if the architecture as a whole wants to be successful and wants to reach the superior goal formulated previously.

6.1.1 Open for Existing Management Infrastructure

Services for teaching will not run in an environment without existing electronic management support. There exist many different applications that are used to manage lectures in today's universities. Some of the existing management infrastructure needs to be used because of organizational requirements, the use of other systems is simply comfortable for the management staff. One reason might be that the staff is simply used to them, perhaps there exist well tried work flows with these existing systems.

Independently of the reasons why the existing systems should be used also in the future, interesting for the new system is only the fact that the old systems will exists further. The question is now, why should the new system be made aware of the old system, why not only develop the new system independently of the old one? This questions should be answered for each old system individually. The relevant criteria for answering this question is: Has the old system informations that are usable, useful or needed by the new system? If this is the case, the new system should be made aware of the old one. If the old system is ignored in such a case, the consequences are the well known one of redundant data: the danger of inconsistencies, the duplicate work for each data acquisition and update to the data. In addition, the integration of a well tested and already introduced system should be in most cases more simple than the development and reimplementation of a new system for the same purpose.

Two examples where the use of an existing system is useful in most cases are the integration of existing student data and access control systems. The data of the students is already present in different systems used in the management of lectures. One example for such a system is HIS used by many universities in Germany. Information about the students is also needed by most teaching services, so an access to the data of the HIS system for those services would be very helpful. For individualized services, students need first to authenticate themselves before they will be allowed to use the service. But students most often have already an account for the use of the computers in the university. The same account could also be used to access the individualized services. If an independent system for the authentication would be used, the system needs to be maintenanced in addition to the existing one.

As these examples show, there are many ways where a new teaching service benefits from existing management infrastructure. But also if the new system would benefit from the data of existing management infrastructure, the attention must also be directed to the privacy and security of the information. It must be ensured that it is allowed to use the data in the new system, and it must be ensured that it is not possible for unauthorized people to access the data.

6.1.2 Open for Existing Application Infrastructure

Different software systems have similar needs for some basic infrastructure services. Examples are services for data storage, for data exchange, logging, security, and user interaction. Web software has some additional needs like services for the communication with the user's

browser and more reliable security, authentication and authorization needs. Because of these common needs of software, especially Web software, different frameworks exist providing these services. New software can simply make use of the infrastructural services provided by these frameworks. The developers are unburdened of the need to reimplement these services once again. The use of these frameworks has many additional benefits. These frameworks are in wide use, an indicator for reliable and thoroughly tested software. Many developers are already used to these frameworks, too. New developers get acquainted with software using these frameworks more quickly than they get acquainted with software that uses a own proprietary framework. Developers that are not already used to one of these frameworks have the chance to get used to the framework comparably fast, because much and good documentation for these frameworks exist.

An example for such a framework, especially for Web enabled software is J2EE [Mic02b]. It provides APIs for most of the needs mentioned above. For the storage of data in relational databases there exist for example different implementations of the JDO [Mic02c] API, like Castor [Gro02a]. Also with the use of EJBs [Mic02a] a persistent storage of data is possible. For the interaction with the user over the Web there exists the Servlet API with implementations like Apache Tomcat [Fou02d]. But besides such large frameworks, which combine many different APIs, also small frameworks exist that have often powerful solutions for one specific need of a new application. An example are different APIs for the processing of XML documents. Some basic APIs for XML processing are already part of the J2EE Specification (like an XML parser API or an XSLT processor API), but until now there exist no J2EE API for the use of Web services. But the Apache project already developed frameworks for the use of Web services with Java: Apache SOAP [Fou02c] and the more recent Apache Axis [Fou02a].

The previous paragraph has only shown a very small amount of existing infrastructural services that exist for Java. Certainly there exist similar services for other programming languages or environments. Which of these frameworks is the best solution for a software project is no easy question. It is not only dependent on the software that should be developed, but also on aspects like the experience of the developers or on the existing software and hardware infrastructure. But it is for sure, that the use of such an application infrastructure is beneficial for a software project.

6.1.3 Open for Existing Standards

Some standards for the exchange of content (LOM, 3.2.1) or the adaptation of the presentation (CSS3 3.3.1, XSLT and DELI, 3.3.2) were already mentioned. Standards allow the interoperability of different systems. They ensure that informations can be exchanged, that different applications can work together. The benefits of using standards are similar to the benefits from using existing application infrastructure: most often good documentation, thoroughly tested, developers already familiar with, etc. One can say that existing application infrastructure set a standard used in the development of new software. But standards like LOM or SOAP [WM02] are useful not only in the development in new individual software, but also in respect to third party applications, other's work in the creation of learning material, and so on. The use of an open standard like SOAP in the communication of the services makes it possible to use existing services with a SOAP interface very simply. Although SOAP is a very new standard, even today there exist already some services accessible through it

(Google Web APIs[Goo02b], IBM UDDI Business Registry [IBM02]).

Beyond the interoperability that is gained if standards are used e.g. for data formats, there is one other reason to use standards: Standards are developed by experts, they discuss different possible solutions and they release a new standard only, if the different interests are fulfilled. This is a long development process, that can be easily made use of: in relying on these standards instead of developing own proprietary solutions, which have often not a quality on par with a standard for the same purpose.

The above shows, that one should rely upon a standard if a standard exist for the need. But not only development costs, interoperability and quality are reasons for the usage of standards. The usage of standards in a product attracts customers because the usage of standards suggest also a high quality of the final product. This observation applied to an architecture for teaching services means, that the architecture will be adopted with a greater probability by possibly interested peoples and organizations if it makes use of well known standards.

6.1.4 Composable with Existing Teaching Applications

In Chapter 4 different systems are presented that provide teaching services. The systems presented are only a fraction of the systems that are used today for teaching services. These systems have approved themselves. There is no reason why these systems should be no longer used. There might be also cases where the old system is used by different people for different lectures, it may be a restriction that the old system needs to be continued, because the new system is not introduced for all lectures. Thus a new system that allows the further use of the old and approved system is preferable over a system that does not allow to use the old one. But there are some prerequisites for the new system if it shall not displace old ones. The new system should integrate the old ones, it should provide the old systems with data where applicable or use the data of the old systems itself. This avoids redundant processing of data.

Similar to an existent management infrastructure, which should be used by a new system as argued in Section 6.1.1, existing teaching applications should be reused, too. The arguments for this goal are also very similar. People might be used to the old systems and the systems have proved to be usable. There are also cases, where the integration of the old system in the new one results in an improvement of the old system. This might be the case if the old system now gets automatically data from the new system instead of relying on manual input. The new one profits from the functionality of the old system. It might be also possible to create a new user interface to the old system, a interface that integrates well with the new system.

An example for such a win-win situation, where both the new system and the old system would profit is an existent AIWBES integrated with a new system. AIWBES like ActiveMath or ELM-ART often have no sufficient user management. Most often there exist no security or special privacy precautions. If such an AIWBES is integrated with an open and composable system that provides security and privacy to the user, both will profit. In addition it would be possible to make use of existent management infrastructure for the authentication of the users. This example shows that the combination of different system may result in true surplus value.

6.1.5 Open for the Use with Existing Teaching Material

There already exists much teaching material. But the greatest problem of each new system with a proprietary format for the teaching material is the lack of this material. This shows that a proprietary format is not a good solution. But also the use of well accepted standards is not the best solution. The compatibility with a standard format allows the use of material that is produced according to the standard, but not the use of material that is produced in proprietary formats, perhaps without the possibly usage within electronic teaching services in mind.

The best solution would be to allow the import and usage of existing teaching material in different formats. It will surely not be possible to use this imported teaching material with all the features that are available with new material produced according to the standard supported by the system. Some adaptive service might be non functional, but at least for a transition phase it would be a good compromise. Sometimes it will be also possible to enhance the imported material manually with meta informations, allowing the use with advanced features that are otherwise only available with material in the standardized format.

With such an import function that allows to add manually some additional meta informations it would be possible to make use of many lecture notes or other teaching material in simple HTML or Postscript format. The acceptance of the system will also benefit if most of the existing material could be reused, at least with some restrictions. This allows a smooth transition phase: not all material needs to be rewritten for the use of the system in a course. Instead the old, already available teaching material can be used until the new standard conforming material is ready, too.

6.1.6 Open for Different Needs

The different applications that are in use today to support teaching show that there are many different needs. The organizational form of different courses can differ substantially. The different organizational forms have different reasons, reasons like the number of participating students, administrative restrictions, and technical ones. This drives the need for an adjustable and a configurable software system.

In one course the students need to submit solutions to exercises every week. A system that wants to support such an organizational form needs a component for the further processing of the solutions. The steps involved are further described in Section 5.5. In another course the student only need to submit solutions once a month. This enables peer reviews as described in Section 5.5.3. This course needs an additional peer review component. In a third course the students do not have to submit solutions at all, instead every tutorium starts with a small quick test on paper. The results of these quick tests should be stored within the system, so a component for the students performance in the quick tests is needed. These examples show how different the needs are already in the single area of exercises.

A monolithic system will not be able to comply to the different needs. Instead only a system of independent services that form a kind of a building set is able to be as adjustable as it needs to be. Instead of a one-size-fits-all approach it is possible with a building set to offer individualized solutions according to the individual needs.

6.2 The Goals and the Existing Architectures

In this section the goals formulated in Section 6.1 are checked against two recent architectures: the architecture developed in the OKI project (Section 4.2.3) and the schema for an architecture for a so-called VU-System [SVS02] as integrative part of the CampusSource project (Section 4.2.4).

These two architectures are chosen for different reasons. One reason was already mentioned: They are recent architectures formulated in the last two years. They have the chance to make use of recent developments of standards, software components, and so on. Other reasons are the goals formulated by the architectures: both have some kind of an integrative ambition, although they are very different. Many of the other system do not have such an integrative ambition. Research systems are often only prototypes that should help to examine the feasibility of some ideas. Other systems only want to solve a single problem. Moreover, both architectures are documented. This makes it possible to discuss, to compare, and to criticize them.

In the next two sections the two architectures are presented, followed by a check of the architectures against the goals formulated in Section 6.1. Where it is necessary, techniques used for the architectures are explained. The third section recapitulates the results of the two previous comparisons and checks. It also mentions the MMiSS (Section 4.4.1) project, which is somewhat related to the CampusSource (Section 4.2.4) initiative.

6.2.1 OKI

In Section 4.2.3 the project was already presented. As mentioned the primary goal of the OKI project is to deliver an architectural specification and implementation for the development of educational applications. These educational applications can access common services through standardized interfaces. This allows to run different educational services from different sources simply on top of a local installation of an implementation of the OKI architecture. The OKI architecture [TSM02] itself takes care of the smooth integration into legacy, back end management infrastructure.

To reach this goal, OKI defines a four level architecture as shown in Figure 5. The four levels are named (from bottom to top):

Institutional Infrastructure: Consists of existing, legacy, back-end information infrastructure. This infrastructure is left untouched by OKI. OKI itself adapts to and make use of this infrastructure where it is feasible and where it is useful. Examples for such an infrastructure are databases, directory services like LDAP, and authentication services like Kerberos [oT02a].

Common Services: Consist of a set of services that are needed by many applications. Services in this layer are file services for the platform independent storage of data, a local unique identifier service for generating unique IDs, and a logging service that can be used for capturing usage information and events for debugging and maintenance. Other services in this layer integrate the institutional infrastructure. This results in services for authentication, in a database service, etc. These integrative services build an abstraction layer for the infrastructural services in providing common interfaces that are

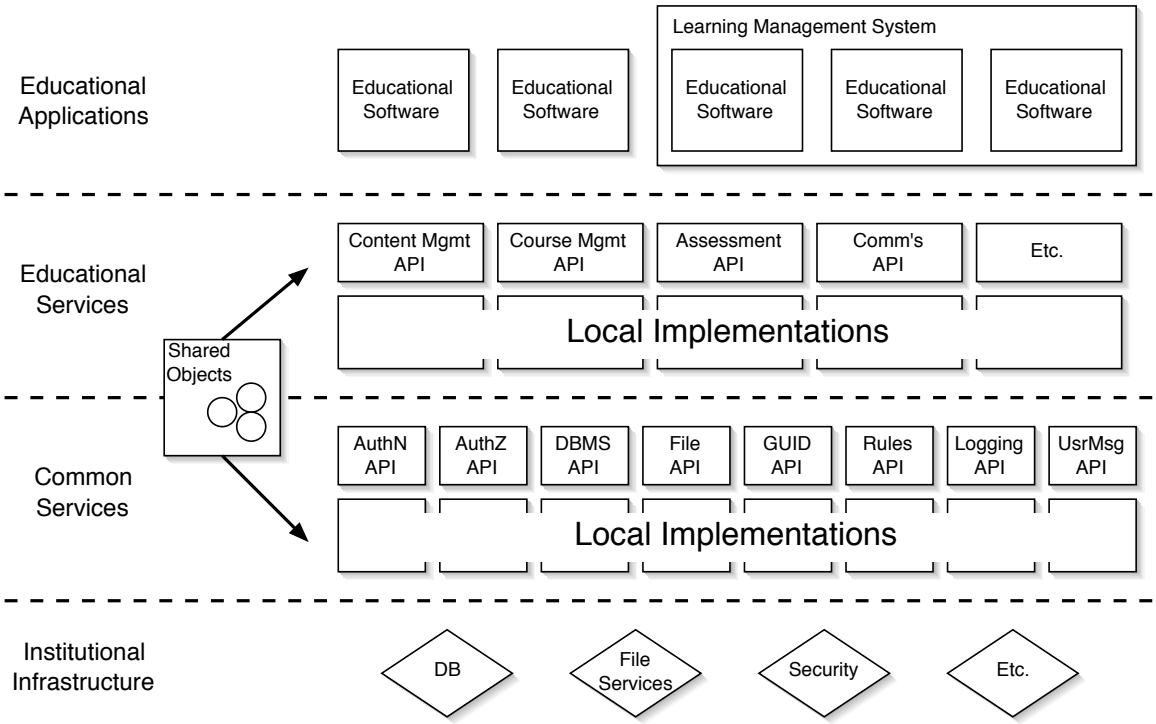

Figure 5: OKI Architecture

independent of the concrete implementations.

Educational Services: Similar to the common services layer this layer consists of set of services that are needed by many applications, too. The difference is, that the services in this layer are more specific to educational applications. Examples for such services are a service for content management, a service for course management, and a service for communication. These services are built completely on top of the services defined in the common services layer. They need not to access any other services outside the common services layer and their own educational services layer.

Educational Applications: The topmost layer consists of educational applications. These applications make use of the services provided by the two lower layers: the common services and the educational services layer. The educational applications have no other dependencies besides the dependencies to these two service layers. This ensures that every educational application that runs on top of one OKI installation will run on top of all OKI installations. The development of such educational applications is not a part of the implementation of the OKI architecture, but it is a part of the OKI project. MIT and Stanford will adapt their current management systems to provide a proof of concept for the OKI architecture.

The services in the common and educational services layer are specified through Java APIs. This conditions an implementation of the interfaces in the Java programming language. Also the educational software on top of these APIs needs to be built in the Java programming

language, or at least it needs to implement a bridge to the programming language used in its implementation on its own.

6.2.1.1 Check against the Goals In the following OKI's architecture is checked against the goals formulated in Section 6.1:

Open for Existing Management Infrastructure: The common services layer of OKI integrates the institutional infrastructure, which includes the management infrastructure. Thus one can say that OKI is open for existing management infrastructure.

Open for Existing Application Infrastructure: OKI defines proprietary APIs for purposes where already widely used solutions exists. No rationales are given by the OKI project why they define own interfaces for purposes like database access, authentication, and logging instead of using existing APIs or other general solutions for these purposes.

Open for Existing Standards: To quote from the article "What is OKI?" [Mer02]:

> Whenever possible this initiative is collaborating with or adopting existing or proposed [...] consortium-developed specifications and standards. OKI is coordinating with existing communities and movements including the Instructional Management System (IMS), the Global Learning Consortium, the work of the Advanced Distributed Learning Network's (ADL) Shareable Courseware Object Reference Model (SCORM) and the Java in Administration Special Interest Group (JA-SIG), to name a few.
>
> [...] It [OKI] will [...] promote specifications and standards around learning management systems architecture. [...] As OKI matures, we hope that it will influence the emerging standards in the LMS space.

This quote shows that OKI not only is willing to use existing standards, but also that it will promote the use of and the development of standards.

Composable with Existing Teaching Applications: Existing teaching applications are not mentioned in the OKI as services that could be integrated. Existing systems needs instead to be adapted to OKI, e.g. the current learning management systems of MIT and Stanford. The OKI project wants to deliver a new product in the market for systems that support teaching. Although they do not want to compete with existing commercial products because of the lack of resources for the support and the promotion of such a product, it wants to influence these products in a way that they become compatible with OKIs architecture. This strategy seems to be successful, as shows a press release of Blackboard [Bla02].

The OKI project thinks that it is possible to influence and to attract other institutions and corporations because of its strong backing. This might be successful in the case of OKI, but it is hard to imagine that any other project might be successful with a similar strategy.

Open for the Use with Existing Teaching Material: As mentioned, OKI supports existing standards, also for teaching material. It is not clear how OKI will support teaching material that does not comply with these standards. Probably this is not in the scope

of the OKI project itself, but in the scope of educational applications developed on top of the OKI services.

Open for Different Needs: Once the OKI services are implemented, different educational software could be used on top of OKI. This software can fulfill different needs. But as long as the APIs for the educational services are not defined it is not clear, if the services provided by these APIs are flexible enough to support the different needs. It is questionable if a single course management API could be powerful enough to support the bunch of different needs for different courses.

The summary of OKI in Section 4.2.3 already lists the pros and cons of the OKI project. It surely will play a big role in the area of systems providing teaching services, but the OKI project left out a great chance by developing an architecture that is not as open and composable as it should be.

6.2.2 CampusSource Architectural Scheme

At the end of Section 4.2.4 the CampusSource architectural scheme was already presented in short. The CampusSource architectural scheme [SVS02] is not like OKI project's a concrete architecture, but more a guideline for the development of new educational software that can be combined with existing applications. To define how the individual applications should be developed is not part of the CampusSource architectural scheme. Instead only some basic requirements are formulated for the individual applications.

Figure 6 shows the architectural scheme of CampusSource. As seen also CampusSource differentiates four layers for its architectural scheme, but they differ almost completely from the four levels of the OKI architecture:

Management Infrastructure: Consists like OKI's institutional infrastructure layer of existing, legacy, back-end infrastructure. As with OKI, this layer is left untouched. The individual components access this layer through the standard interfaces provided by the legacy software.

Local Data Storage Layer: The local data storage layer is the first of the three common layers of information systems [Fow97]. It corresponds to the data source layer in the three tier architecture. This layer stores all data that is only used by the individual application itself.

Application Layer: The application layer combines the domain layer and the application logic part of the application layer of the common three tier architecture. What is special to the CampusSource architectural scheme is the CORBA [Gro02c] interface at this layer. CORBA is a mainly language neutral standard for the exchange and the performing of messages between and by objects. This interface allows the individual applications to cooperate with and make use of each other.

Web Layer: The Web layer is the presentation part of the application layer in the common three tier architecture. Because it is an architectural scheme for Web applications, this layer consists of a Web interface for each application. Special to the CampusSource architectural scheme is, that there exists an integration application, that access the

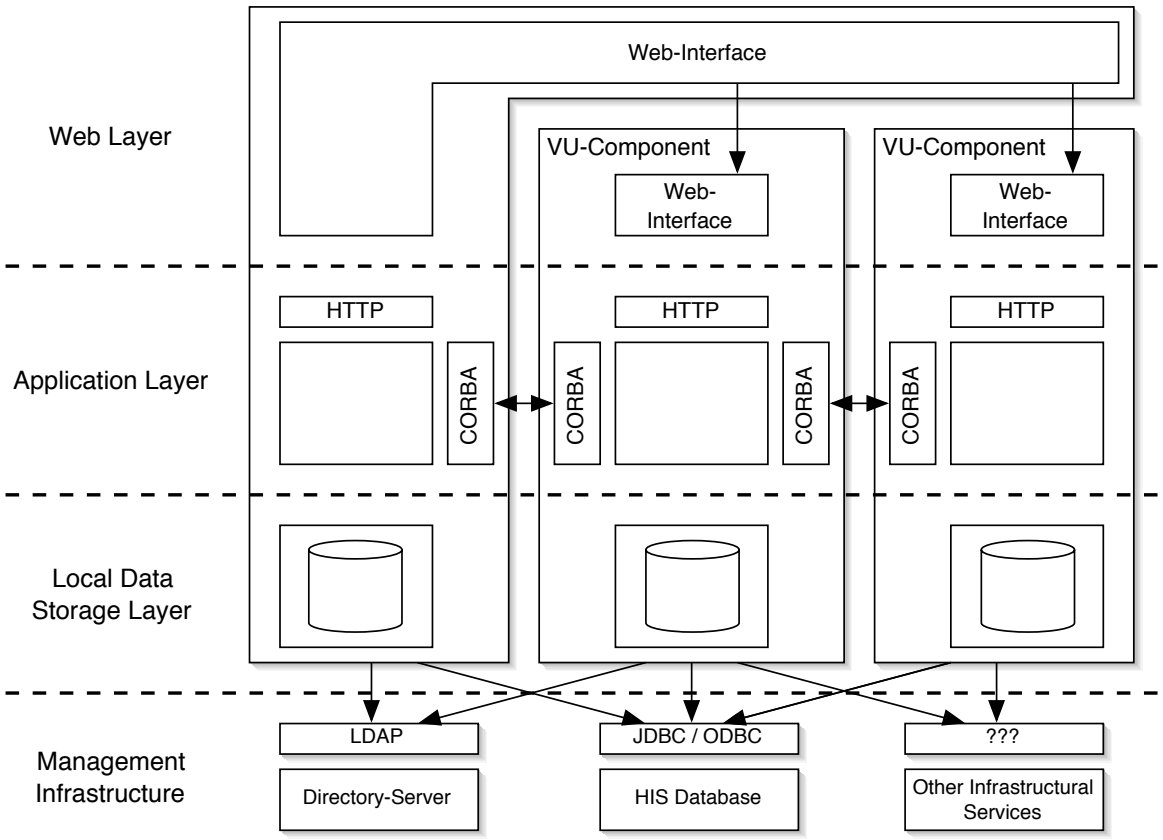

Figure 6: CampusSource Architectural Scheme

other applications not only through CORBA interfaces at application layer level, but also through the Web interface. This allows the integration application to provide a common user interface to the different applications

As the architectural scheme shows, CampusSource is very different from OKI. CampusSource tries to make use of as many existing applications as possible. This is done by requiring only two things: a Web and a CORBA interface. This also shows the very high level of the architectural scheme of CampusSource.

6.2.2.1 Check Against the Goals In the following CampusSource's architectural scheme is checked against the goals formulated in Section 6.1:

Composable with Existing Teaching Applications: The core of the architectural scheme is the composition of existing teaching applications. The existing applications communicate through CORBA and their user interfaces are combined by the integration component.

Open for Different Needs: CampusSource combines whole applications, not individual components. The combination of whole applications may be not flexible enough. The applications themselves need to be so configurable, that a configuration in respect to the individual needs is possible.

Other Goals: The goals (open for existing management infrastructure, open for existing application infrastructure, open for existing standards, and open for the use with existing teaching material) are not addressed by the the CampusSource architectural scheme. These goals can be reached, but it is the task of the individual applications. For example, the individual applications are themselves responsible to make use of the existing management infrastructure. The architecture scheme defines no special way how to make use of the legacy applications.

The idea of simply combine existing applications is very interesting, but it is the question if it is possible to build a really useful system by this method. The check against the goals of an open and composable architecture for teaching services shows, that it lacks many things, they are simply left to the individual applications.

6.2.3 Summary

The two architectures presented show two possible ways how an integrative system could look like. They are both very extreme architectures: OKI wants to build a new, better system without taking existing applications into account, while CampusSource solely build upon existing systems and provides only a scheme, which is not even a framework.

But the two also differ in their way the different components or applications that build the whole system communicate with each other. While OKI prefers the use of an Java API, a very simple and easy way for the programmers, CampusSource prefers the more complex and more programming language independent way of using CORBA. But there exists alternatives to these two ways: ActiveMath (Section 4.1.1) use XML-RPC to combine the

different applications that are used in the system. This is a even more language independent and thus open way to combine different applications than CORBA. MMiSS (Section 4.4.1) probably also use XML-RPC for the integration of WebAssign and ActiveMath components to a new system. ActiveMath is also otherwise interesting because it combines components and applications with a size and complexity between the full blown applications of CampusSource and the relatively small APIs of OKI.

The concept behind ActiveMath and probably also behind MMiSS seems more promising than the concepts of OKI and CampusSource. But ActiveMath is only an AIWBES, it integrates only components for the purpose of creating a learning environment for mathematics. MMiSS on the other hand is currently in an early conceptual phase, it is not clear how the integration of WebAssign and ActiveMath will be done.

6.3 Proposal of an Open Architecture

In this section an architecture is proposed, that takes up and combines some of the ideas of the architectures presented in Section 6.2 while trying to avoid their shortcomings.

First a high level overview for the proposed architecture is given. This first look is followed by a more in depth discussion of the major components and their possible implementations. In the last section the rationales that resulted in this architecture are discussed. The differences between this architecture and the architectures already presented will be emphasized.

6.3.1 A First Look at the Architecture for Open and Composable Teaching Services

Figure 7 shows the scheme of the architecture for open and composable teaching services. There exist two major layers: the service provider layer and the integration layer, a kind of service request layer.

For some of the service providers Web service adapters are used to allow a more tight integration with the other service providers and the integration layer. The service providers are divided in the two major groups of services with a user interface and back-end services. The services with a user interface are simply existing educational applications or applications and services that are useful in the educational context. The group of back-end services consists mainly of existing management infrastructure. This infrastructure is made use of by the other services or by the integration layer, but is not of direct interest to the end user.

The integration layer itself is liable for the interaction with the user. This layer processes the requests from the user and builds the response. It assigns the service providers that are relevant to the user's request to process the request. The responses of the service providers are collected by the integration layer and the individual responses are combined to an integrated one.

6.3.2 A More In-Depth Look on the Architecture

The previous section provided already a first look on the architecture. The question how the individual components work and how they work together is left open until now. This question

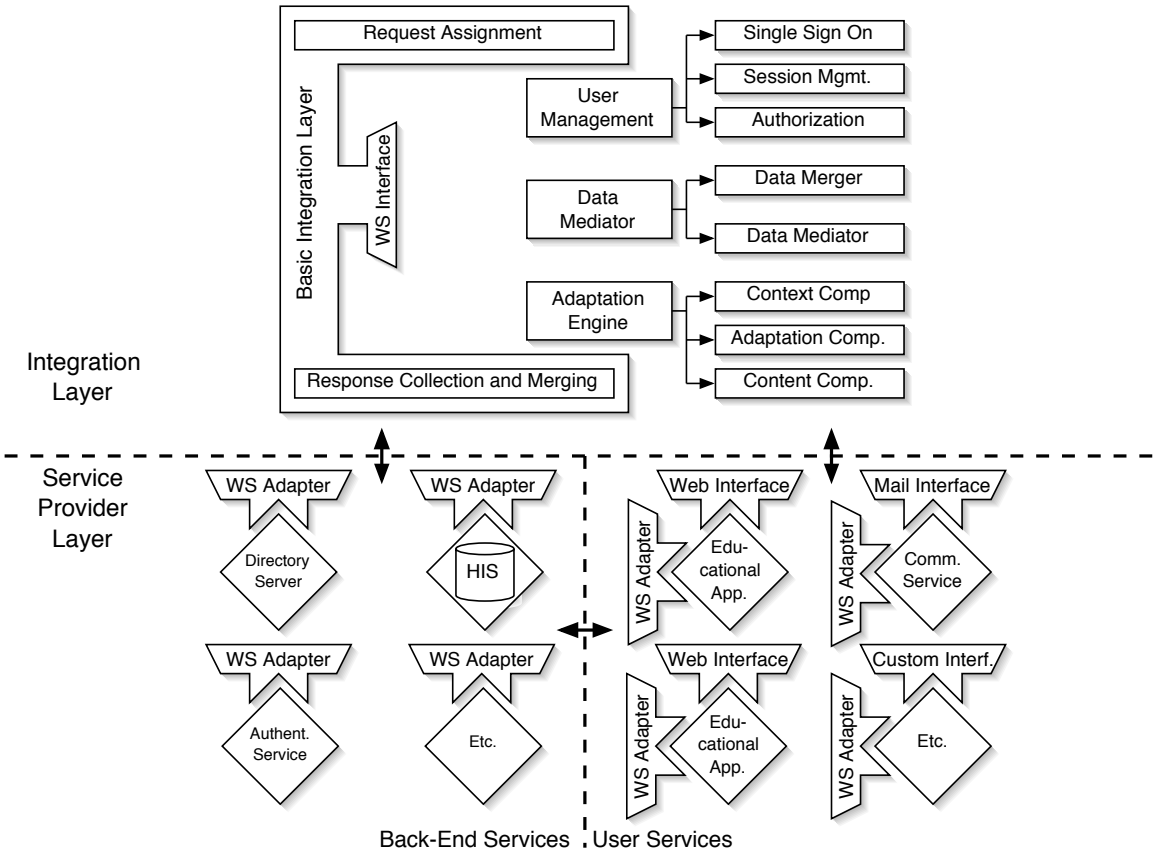

Figure 7: Architecture for Open and Composable Teaching Services

will be answered in the following two sections. The two major layers, the service provider layer and the integration layer, will be explained in more depth.

6.3.2.1 Service Providers: As already mentioned, the service providers consist of two major groups, existing management infrastructure and educational applications. Both groups are on the same level in this architecture and they are also handled very similar. They are existing applications or systems and are left untouched. Instead of changing the systems themselves, on top of the systems Web service adapters are created that translate from the legacy interfaces of the systems to a more open and also more flexible interface.

With the help of Web service adapters such an open and flexible interface is possible. Web services are a new approach to the inter-operation of different applications. The vision of Web services is, that applications are enabled to interact among themselves over the Web, just like humans today interact with Web applications. To quote the definition of W3C's Web service architecture working group [W3C02c]:

> A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols

Because of the text based XML messaging, a Web service can be written in any programming language and run in any environment. Standardized and widely available protocols like HTTP or SMTP can be used as transport protocol .

The advantages of this Web services approach are, that it is possible to maintain the installations of the educational applications and the management infrastructure unchanged. They are equipped with an additional Web services interface, which allows to use the service providers together, whether or not they are running on different servers, are implemented in different programming languages or running in different application environments. The very loose coupling, only based on the exchange of text messages, allows the exchange and upgrade of single services. This can be done without affecting other service that make use of them.

But there is also a drawback to this approach. Given that Web service are a new approach it is no mature technology. The specifications (like SOAP [WM02]) are evolving very fast and also their implementations (like Apache Axis [Fou02a]) for the different programming environments are updated very frequently.

It should be mentioned that the architecture of ActiveMath (Section 4.1.1) also makes use of a similar approach to connect the external mathematical applications with the ActiveMath system. XML-RPC [Sof02] is used to implement this connection. This shows that such a loose binding of components also makes sense for the creation of educational applications. If new educational applications make use of such a flexible architecture of individual components, also these components can be reused by other applications. This results in a building set of components that is needed to fulfill the very different needs of different courses or other usage scenarios for the whole system.

6.3.2.2 Integration Layer: The integration layer can be seen as the core of the open architecture for composable teaching services. One can argue, that this layer can be very

thin. That it only needs to combine the user interfaces of the individual service providers in a portal like way. And yes, that is true. A integration layer implemented as such a portal can be implemented relatively quick, e.g. with the help of existing frameworks for the creation of so-called enterprise information portals like Apache Jetspeed [Fou02b]. But although this will work, the end users will not profit very much from such a solution. Such a solution is only the first step in the creation of an integration layer of the architecture described here.

A more advanced integration layer consists of different components that can work almost independently of each other. These components are in fact simply special service providers. They offer services that are used primarily by the integration layer. In most cases the other service providers communicate only with the integration layer as intermediatory entity with these special services. Following is a list of essential services that an integration layer should offer.

User Management: In a first step user management offers a single sign-on service. This means that the user do not need to log in to each of the integrated applications separately. Instead he only needs to authenticate himself to the integration layer, which automatically performs the authentications to the other applications as needed. For this purpose the single sign-on service can make use of existing infrastructure for authentication. Depending on the authentication mechanisms used by the integrated applications, this service needs access to the different authentication mechanisms or it needs to store the users' access informations itself.

Once a single sign-on mechanism exists, also a session management needs to be implemented. The integration layer must ensure that all sessions of the user have been ended when he ends the session with the integration layer. Not implementing such a session management is a possible security risk. Intruders may use open but unused sessions to gain access to the system.

The integration layer also can implement a kind of access control to different services. If the integration layer is aware of the group, like student or staff, the user belongs to, it can integrate different services or prevent the access to some of them. It can act as an authorization service. A system that is aware of the group the user belongs to is the first step of a context aware system.

Adaptation Engine: The adaptation engine is the prerequisite for offering adaptive features. It should be noted that the integration layer does not affect adaptive features of the integrated application. An AIWBES offer the same adaptive features whether or not it is integrated.

But there are additional adaptive features that can be offered by the integration layer. One example is the adaptation of the presentation to different capabilities of the client device. Other adaptive possibilities results from the integration of different services. The services can be combined depending on the current context. If a service for communication features exists, the integration layer may offer these features in conjunction with the other services where appropriate.

An implementation of such an adaptation engine will orientate itself on the conceptual view in Figure 3. For the implementation of the adaptation engine techniques and systems like DELI (Section 3.3.2) for a server side solution or Style Sheet Selectors (Section 3.1.5) for a client side solution may be used.

Data Mediator: Another task of the integration layer is to be a mediator for the data of the different service providers. The applications that are the service providers usually have their own data storage. If two or more applications need the same information, they should be configured to access only one data storage to avoid redundant data. But if the applications are not flexible enough to allow the use of a common data storage, another solution is needed. The integration layer acting as data mediator can be such a solution.

There are two scenarios conceivable: all updates to the data are triggered by requests of the integration layer, or the existing applications are used also independently and the data can change without notification of the integration layer. In the first case, the integration layer simply needs to check for the changes after delegating the request to the corresponding service and update the data of the other services either directly or through the provided interfaces. The second case is more complicated. The integration layer needs to check for updates every time when a service is requested that might access outdated data. Because this could be very time intensive, it should be decided individually in every case if this is the best solution. Perhaps a data merge once a day is sufficient, too.

For a modular integration layer it is also possible to place some components on the client and some on the server side. If the components of the integration layer uses Web service interfaces, these interfaces are accessible over the Internet. Both, the user machine with a thick client and a server with a thin client on the other side, can make use of them. If the presentation generation components of the integration layer are placed on the client side, a richer and more responsible user interface is possible than with a thin client with HTML as presentation language. Sensible user data can also be stored on the client machine. Parts of the adaptation engine located at the client side makes sending security and privacy relevant data over the Internet unnecessary. But these advantages should be weighted against the well known drawbacks of thick clients: They need to be delivered to and maintained on every client machine, while a thin client only uses a Web browser, which is installed on every desktop computer.

6.3.3 Rationales for this Architecture

To form a common Web interface for different applications and information sources is a problem not specific to the domain of teaching services. A company with its grown infrastructure of electronic applications and its needs to combine the different information sources in making this infrastructure accessible over the company's intranet has a very similar problem. Solutions to these needs are called Enterprise Application Integration (EAI) servers and Enterprise Information Portals (EIP). The EAI server works as a junction for all informations. It mediates between the different applications, it stores information useful for the applications, and it hands down user requests to the applications. If an EIP is used in conjunction with an EAI server, it is usually nothing more than a Web interface to the EAI. The open architecture for composable teaching services has similarities to solutions in the EAI and EIP domain, where similar problems were already solved many times.

6.3.3.1 Check against the Goals Also this architecture needs to be checked against the goals that are formulated in Section 6.1:

Open for Existing Management Infrastructure Existing infrastructure can be made use of either directly or through adapters. These adapters may also implement an abstraction layer which makes it possible to exchange the underlying implementation.

Open for Existing Application Infrastructure Existing application infrastructure may be used extensively by the implementation of the integration layer. It may be implemented as a Java Servlet application and it can even make use of a J2EE Application Server [Mic02b] for runtime services like messaging and data persistence. Toolkits like Apache Axis [Fou02a] exist, that makes it easy to make use of and to offer Web services.

Open for Existing Standards The use of Web service standards for the integration of existing and new services shows the use of standards in this application. The use of standards in the area of learning material is the scope of the service providing applications and also the scope of the specifications of the adapters. The adapters may translate between proprietary formats and open standards.

Composable with Existing Teaching Applications The architecture is about using existing applications. But these applications are not only made usable for a single new system. The usage of Web service protocols for the integration of the applications makes them usable also for other systems.

Open for the Use with Existing Teaching Material Similar to the goal of being open for existing standards, this is the scope of the service providing applications. But every teaching material that is accessible by the integration layer may be used with a corresponding adaptation service.

Open for Different Needs The loose composition of existing applications, existing infrastructure, and also new services through Web service protocols makes it possible to configure the resulting system according to individual needs. Components may be exchanged, added, or omitted according to the different needs.

This architecture adopts the idea of CampusSource's architectural scheme of combining existing applications. But here these applications are just simple service providers that happen to have also a own user interface. Other service providers may exist on the same level. They support the integration layer in its tasks. This results in a more flexible and also a better extensible architecture. One can just begin with a simple integration layer and add additional service providers step by step. The system will be usable at each step. This is in contrast to the idea of OKI. OKI needs to complete the definition of the APIs. After this the implementation of the APIs and the adaptation of educational applications can start. The system is usable only if all of these three steps are completed.

7 Conclusion

Contextual Web Services for Teaching is objectively not a single topic. There are different research areas relevant. For the creation of adaptive, context aware services different strategies are possible. Some of them are inspired by the need to adapt the information to the capabilities of the client's device, others by the need to adapt to the knowledge of the users. Today's applications that provide teaching services differ substantially. Reasons are different needs, different goals, different resources for the implementation, etc. The tentative classification of teaching services shows, that there exist a range of services that are not only relevant for teaching purposes, and the rationales for the proposed architecture refer to similar architectures for enterprises.

Although there exist so many different but relevant things, the papers, articles and Web sites about one topic seems to ignore other also relevant topics very often. An example is the Web site of the OKI project where AIWBES are not even mentioned. But also other Web sites on course management systems seems to ignore the existing AIWBES. Papers about AIWBES on the other hand almost completely ignore the course management systems. References to course management systems from AIWBES are most often only disparaging valuations of there adaptive features. The privacy and security support, or the ease of use for technically little educated people are often ignored. Even more interesting is that the solutions for the adaptation of content for different client devices and the solutions used in AIWBES or most other adaptive hypermedia systems seem to be developed independently, without much cooperation. This is surprising, because the solutions needed are almost the same.

This thesis has presented the different areas, solutions, applications, etc. that are relevant to the topic of contextual Web services for teaching. Hopefully this thesis with the architecture presented in Section 6.3 would contribute to a more integrative and cooperative view.

References

[AHA02] AHA! Adaptive Hypermedia for All, 2002. http://aha.win.tue.nl/.

[And02] Andamooka. Open Support for Open Content, 2002. http://www.andamooka.org/.

[BBGC00] A. J. Bernheim Brush, David Bargeron, Anoop Gupta, and JJ Cadiz. Robust Annotation Positioning in Digital Documents . Technical Report MSR-TR-2000-95, Collaboration and Multimedia Systems Group, Microsoft Research, Microsoft Corporation, Redmond, US, 2000. http://www.research.microsoft.com/research/coet/Annotations/TRs/00-95.pdf.

[BG00] Jesus G. Boticario and Elena Gaudioso. A Multiagent Architecture for a Web-Based Adaptive Educational System . Technical Report TR SS-00-01, 2000. http://www.ia.uned.es/~jgb/publica/download/aaai00-s.ps.

[BGC01] Jesus G. Boticario, Elena Gaudioso, and Carlos Catalina. Towards Personalized Learning Communities on the Web . In *European Conference on Computer-Supported Collaborative Learning 2001, Euro-CSCL 2001, Maastricht*, 2001. http://www.mmi.unimaas.nl/euro-cscl/Papers/56.pdf.

[BHW99] Paul De Bra, Geert-Jan Houben, and Hongjing Wu. AHAM: A Dexter-Based Reference Model for Adaptive Hypermedia . In *10th ACM Conference on Hypertext and Hypermedia, Darmstadt, Germany*, 1999. http://wwwis.win.tue.nl/~debra/ht99/ht99.ps.

[BK02a] François Bry and Michael Kraus. Perspectives for Electronic Books in the World Wide Web Age . *The Electronic Library*, 20(4), 2002. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2002-6.

[BK02b] François Bry and Michael Kraus. Adaptive Hypermedia made simple using HTML/XML Style Sheet Selectors. In *2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web Based Systems, Malaga, ES*, 2002. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2002-1.

[BK02c] François Bry and Michael Kraus. Position Paper: Style Sheets for Context Adaptation. In *W3C Workshop on Delivery Context*, 2002. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2002-3.

[Bla02] Blackboard. Blackboard Announces Adoption Strategy for Open Knowledge Initiative (OKI) Specifications, 2002. http://company.blackboard.com/press/viewrelease.cgi?tid=208.

[BM01] Peter Brusilovsky and Philip Miller. Course Delivery Systems for the Virtual University . In *Access to Knowledge: New Information Technologies and the Emergence of the Virtual University*, pages 167–206, 2001. http://www2.sis.pitt.edu/~peterb/papers/UNU.pdf.

[Bre02] Jeff Breidenbach. The Mail Archive, 2002. http://www.mail-archive.com.

[Bru96] Peter Brusilovsky. Methods and Techniques of Adaptive Hypermedia . *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996. http://www.contrib.andrew.cmu.edu/~plb/UMUAI.ps.

[Bru01] Peter Brusilovsky. Adaptive Hypermedia . *User Modeling and User-Adapted Interaction*, (11):87–110, 2001. http://umuai.informatik.uni-essen.de/brusilovsky-umuai-2001.pdf.

[But02] Mark H. Butler. Using Capability Classes to Classify and Match CC/PP and UAProf Profiles . Technical Report HPL-2002-89, HP Labs Bristol, 2002. http://www-uk.hpl.hp.com/people/marbut/capClass.htm.

[Cor02] Borland Software Corporation. Interbase, 2002. http://www.borland.com/interbase.

[Cun02] Ward Cunningham. Portland Pattern Repository and WikiWikiWeb Frontpage, 2002. http://c2.com/cgi/wiki.

[DBR01] Sarah Drummond, Cornelia Boldyreff, and Magnus Ramage. Evaluating Groupware Support for Software Engineering Students . *Computer Science Education*, 11(1):33–55, 2001. http://www.dur.ac.uk/sarah.drummond/papers/CSEJournal2000.doc.

[dJHRCV01] Guillermo de Jesus Hoyos Rivera, Jean-Pierre Courtiat, and Thierry Villemur. A Design Framework for Collaborative Browsing . In *IEEE 10th Intl. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Massachusetts Institute of Technology, Cambridge*, 2001. http://www.dsi.unimo.it/wetice2001/courtiat.pdf.

[DMKSH89] H. Dieterich, U. Malinowski, T. Kühme, and M. Schneider-Hufschmidt. State of the Art in Adaptive User Interfaces . In *Adaptive User Interfaces: Principles and Practice*, pages 13–48. North-Holland, 1989. .

[DMT02] DMTCS. Discrete Mathematics and Theoretical Computer Science, 2002. http://dmtcs.loria.fr/.

[Edu02] Edutech. Comparison of Web Based Course Environments, 2002. http://www.edutech.ch/edutech/tools/comparison_e.asp.

[For01] WAP Forum. User Agent Profile, Mar 2001. http://www1.wapforum.org/tech/documents/WAP-248-UAProf-20010530-p.pdf.

[Fou02a] Apache Software Foundation. Apache Axis, 2002. http://xml.apache.org/axis.

[Fou02b] Apache Software Foundation. Apache Jetspeed Enterprise Information Portal, 2002. http://jakarta.apache.org/jetspeed.

[Fou02c] Apache Software Foundation. Apache SOAP, 2002. http://xml.apache.org/soap.

[Fou02d] Apache Software Foundation. Apache Tomcat Servlet Container, 2002. http://jakarta.apache.org/tomcat.

[Fow97] Martin Fowler. *Analysis Patterns: Reusable Object Models* . Object Technology Series. Addison Wesley, 1997. .

[GBS00] Hans-W. Gellersen, Michael Beigl, and Albrecht Schmidt. Sensor-Based Context-Awareness for Situated Computing . In *Workshop on Software Engineering for Wearable and Pervasive Computing, SEWPC00, 22nd International Conferencn on Software Engineering, ICSE 2000, Limerick, Ireland*, 2000. http://www.teco.edu/~michael/publication/gellersen_sewpc00_sensor-based-context.pdf.

[GEL02] Judith Gal-Ezer and David Lupo. Integrating Internet Tools Into Traditional CS Distance Education: Students' Attitudes . *Computers and Education*, 38(4):319–329, 2002. http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VCJ-45CN1DK-2-C&_cdi=5956&_orig=browse&_coverDate=05%2F31%2F2002&_sk=999619995&wchp=dGLStV-lSzBV&_acct=C000032323&_version=1&_userid=616146&md5=1e6eeffe2695aec68b9790cc62bf6ffc&ie=f.pdf.

[Gmb02] HIS GmbH. HIS: Hochschul-Informations-System, 2002. http://www.his.de.

[Goo02a] Google. Google Groups: Post and read comments in Usenet discussion forums, 2002. http://groups.google.com.

[Goo02b] Google. Google Web APIs, 2002. http://www.google.com/apis.

[Gro02a] ExoLab Group. Castor: Open Source Data Binding Framework for Java, 2002. http://castor.exolab.org.

[Gro02b] JBoss Group. JBoss Application Server, 2002. http://www.enhydra.org.

[Gro02c] Object Management Group. CORBA: Common Object Request Broker Architecture, 2002. http://www.corba.com.

[Gro02d] The AIMS Group. MARC: Mailing list ARChives at AIMS, 2002. http://marc.theaimsgroup.com.

[GS01] Tom Gross and Marcus Specht. Awareness in Context-Aware Information Systems . In *Mensch und Computer, 1. Fachübergreifende Konferenz, Bad Honnef, DE*, 2001. http://orgwis.gmd.de/%7Egross/publ/mc2001/gross_specht_mc2001.pdf.

[Ham89] N. Hammond. Hypermedia and Learning: Who Guides Whom? . *Lecture Notes in Computer Science*, 360:167–181, 1989. .

[HL02] HP-Labs. DELI: A Delivery Context Library For CC/PP and UAProf, 2002. http://delicon.sourceforge.net/.

[HN00] Johan Hjelm and Mikael Nilsson. Towards RDF-Based Profiles for Context-Based Position-Dependent Services . In *WAP Forum, W3C Workshop on Position Dependent Information Services*, 2000. http://www.w3.org/Mobile/posdep/ericsson-position-paper.htm.

[HN02] Nicola Henze and Wolfgang Nejdl. Knowledge Modeling for Open Adaptive Hypermedia . In *2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, Malaga, ES*, 2002. http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2002/ah2002/henze_ah02.ps.

[IBM02] IBM. IBM UDDI Business Registry, 2002. https://uddi.ibm.com/ubr/registry.html.

[IEE02] IEEE. Learning Object Metadata, 2002. http://ltsc.ieee.org/doc/wg12/LOM_WD6-1_1_without_tracking.pdf.

[ILI02] ILIAS. Integriertes Lern-, Informations- und Arbeitskooperations-System, 2002. http://www.ilias.uni-koeln.de/.

[IMS01a] IMS. Content Packaging Information Model, 2001. http://www.imsglobal.org/content/packaging/.

[IMS01b] IMS. Learner Information Packaging Information Model Specification, 2001. http://www.imsglobal.org/profiles/lipinfo01.html.

[IMS02] IMS. IMS Global Learning Consortium, 2002. http://www.imsproject.org/.

[JES02] JESS. Rule Engine for the Java Platform, 2002. http://herzberg.ca.sandia.gov/jess/.

[KKP01] Alfred Kobsa, Jürgen Koenemann, and Wolfgang Pohl. Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships . *The Knowledge Engineering Review*, 16(2):111–155, 2001. http://www.ics.uci.edu/~kobsa/papers/2001-KER-kobsa.pdf.

[Kob01] Alfred Kobsa. Generic User Modeling Systems . *User Modeling and User-Adapted Interaction*, (11):49–63, 2001. http://www.ics.uci.edu/%7Ekobsa/papers/2001-UMUAI-kobsa.pdf.

[Kob02] Alfred Kobsa. Personalized Hypermedia and International Privacy . *Communications of the ACM*, 45(5):64–67, 2002. http://www.ics.uci.edu/%7Ekobsa/papers/2002-CACM-kobsa.pdf.

[Koh00] Michael Kohlhase. OMDoc: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge . In *5th International Conference Artificial Intelligence and Symbolic Computation: Theory, Implementations and Applications, AISC'2000, Madrid, ES*, 2000. http://www.cs.cmu.edu/~kohlhase/papers/aisc.ps.

[KP95] Alfred Kobsa and Wolfgang Pohl. The User Modeling Shell System BGP-MS . *User Modeling and User-Adapted Interaction*, 4(2), 1995. http://zeus.gmd.de/~kobsa/papers/1995-UMUAI-kobsa.ps.

[Lan02] Bruce Landon. Online Educational Delivery Applications: A Web Tool for Comparative Analysis, 2002. http://www.c2t2.de/landonline/.

[MAB+01] Erica Melis, Eric Andres, Jochen Büdenbender, Adrian Frischauf, George Goguadze, Paul Libbrecht, Martin Pollet, and Carsten Ullrich. ActiveMath: A Generic and Adaptive Web-Based Learning Environment . *Artificial Intelligence in Education*, 12(4), 2001. http://www.ags.uni-sb.de/%7Eilo/articles/GenericAndAdaptiveWebBasedLearningEnvironment.pdf.

[Mer02] Jeff Merriman. What is OKI?, 2002. http://web.mit.edu/oki/product/whtpapers/whatis.html.

[Mic02a] Sun Microsystems. Enterprise JavaBeans'Technology, 2002. http://java.sun.com/products/ejb/.

[Mic02b] Sun Microsystems. Java 2 Platform Enterprise Edition, 2002. http://java.sun.com/j2ee.

[Mic02c] Sun Microsystems. Java Data Objects, 2002. http://access1.sun.com/jdo/.

[Mic02d] Sun Microsystems. Java Servlet Technology, 2002. http://java.sun.com/products/servlet/.

[Mül02] Thomas Müller. Hypersonic SQL Open Source Java Database, 2002. http://hsqldb.sourceforge.net/internet/hSql.html.

[Obj02a] ObjectWeb. Enhydra Application Server, 2002. http://www.enhydra.org.

[Obj02b] ObjectWeb. Java Open Application Server, 2002. http://www.objectweb.org/jonas.

[Ope02] OpenUSS. Open University Support System, 2002. http://openuss.sf.net/.

[oT02a] Massachusetts Institute of Technology. Kerberos: The Network Authentication Protocol, 2002. http://web.mit.edu/kerberos/www/.

[oT02b] Massachusetts Institute of Technology. OKI: Open Knowledge Initiative, 2002. http://web.mit.edu/oki.

[PMP01] Lutz Prechelt, Guido Mahlpohl, and Michael Philippsen. Finding Plagiarisms Among a Set of Programms with JPlag . 2001. http://www2.informatik.uni-erlangen.de/~phlipp/mypapers/jplag_jucs2001.pdf.

[PW00] Thomas A. Phelps and Robert Wilensky. Robust Intra-Document Locations . In *9th International World Wide Web Conference, WWW9, Amsterdam, NL*, 2000. http://www9.org/w9cdrom/312/312.html.

[RR201] RR2000. Interaktives Lernprogramm zur Rechtschreibung, 2001. http://apsymac33.uni-trier.de:8080/elm-art/RR2000-login-d.html.

[Sch01] Georg Schneemayer. Einsatz elektronischer Medien für den Übungsbetrieb, 2001. http://www.pms.informatik.uni-muenchen.de/publikationen/projektarbeiten/Georg.Schneemayer/ausarbeitung.ps.gz.

[Sie02] Sabine Siekmann. Which Web Course Management System is Right for Me? A Comparision of WebCT3.1 and Blackboard 5.0, 2002. http://astro.temple.edu/~jburston/CALICO/review/webct-bb.htm.

[SO00] Marcus Specht and Reinhard Oppermann. ACE - Adaptive Courseware Environment . *Lecture Notes in Computer Science*, 1892:380–??, 2000. http://fit.gmd.de/~oppi/publications/NRHM.pdf.

[Soc02] The OpenMath Society. Homepage, 2002. http://www.openmath.org/.

[Sof02] UserLand Software. XML-RPC Home Page, 2002. http://www.xmlrpc.com.

[SPS+98] C. Stephanidis, A. Paramythis, M. Sfyrakis, A. Stergiou, N. Maou, A. Leventis, G. Paparoulis, and C. Karagiannidis. Adaptable and Adaptive User Interfaces for Disabled Users in the AVANTI Project . 1430:153–166, 1998. .

[SVS02] H.-W. Six, J. Voss, and W. Schäfer. Architekturschema für VU-Systeme, 2002. http://www.campussource.de/projekte/architektur/architektur_vusysteme.html.

[TNP97] Michael B. Twidale, David M. Nichols, and Chris D. Paice. Browsing is a Collaborative Process . *Information Processing and Management*, 33(6):761–783, 1997. http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VC8-3SX2J1V-6-1&_cdi=5948&_orig=browse&_coverDate=11%2F30%2F1997&_sk=999669993&wchp=dGLbVlb-lSztb&_acct=C000032323&_version=1&_userid=616146&md5=26b6b6300097f192b4f579fd4303e6c1&ie=f.pdf.

[TSM02] Scott Thorne, Chuck Shubert, and Jeff Merriman. OKI Architecture Overview, 2002. http://web.mit.edu/oki/product/whtpapers/arch_overview.html.

[VVD01] Arja Veerman and Else Veldhuis-Diermanse. Collaborative Learning Through Computer-Mediated Communication In Academic Education . In *European Conference on Computer-Supported Collaborative Learning 2001, Euro-CSCL 2001, Maastricht*, 2001. http://www.mmi.unimaas.nl/euro-cscl/Papers/166.doc.

[W3C00] W3C. Composite Capabilities/Preference Profiles: Requirements and Architecture, Jul 2000. http://www.w3.org/TR/CCPP-ra/.

[W3C02a] W3C. Annotea Mailing List, 2002. http://lists.w3.org/Archives/Public/www-annotation/.

[W3C02b] W3C. Media Queries, Jul 2002. http://www.w3.org/TR/css3-mediaqueries.

[W3C02c] W3C. W3C Web Services Architecture Working Group, 2002. http://www.w3.org/2002/ws/arch/.

[W3C02d] W3C. WWW Mobile Mailing List, 2002. http://lists.w3.org/Archives/Public/www-mobile/.

[WA99] Wein-ART. Interaktives Lernprogramm zum Wein in Deutschland, 1999. http://Apsymac33.uni-trier.de:8080/Wein-ART.

[WB01] Gerhard Weber and Peter Brusilovsky. ELM-ART: An Adaptive Versatile System for Web-based Instruction . *International Journal of Artificial Intelligence in Education*, (12), 2001. http://www2.sis.pitt.edu/~peterb/papers/JAIEDFinal.pdf.

[WKW01] *Developing Adaptive Internet Based Courses with the Authoring System NetCoach* , volume 2266 of *Lecture Notes in Computer Science*, 2001. http://home.ph-freiburg.de/weibelza/literatur/weber-ah2001.pdf.

[WM02] W3C and Nilo Mitra. SOAP Version 1.2 Part 0: Primer, Jun 2002. http://www.w3.org/TR/2002/WD-soap12-part0-20020626/.

[YHK95] W. Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol (RFC 1777), 1995. http://www.ietf.org/rfc/rfc1777.txt.