

Using Highly Sophisticated Middleware For Building Arbitrarily Distributed Teaching Environments

Christof Dallermassl, Heimo Haub, Harald Krottmaier,
Klaus Schmaranz, Philipp Zambelli
{cdaller,hhaub,hkrott,kschmar,pzamb}@iicm.edu

**Institute for Information Processing and
Computer Supported new Media**

This paper deals with the development of highly sophisticated teaching environments. We took a look at the requirements that such a system has to fulfill to meet the needs of our modern society and to remain easily adaptable to forthcoming, new technologies. The results of our research show that the cost for design and implementation of a distributed teaching environment can be dramatically reduced by using a middleware system. We thereto present the concepts of the Dinopolis middleware system which is highly modular and extendable and show how it may be used as the basis for the development of teaching environments. Using the Dinopolis middleware system not only eases the process of development but also guarantees that new technologies are becoming available without any further effort.

Keywords: Distributed Teaching Environment, Middleware, Application Framework, Distributed Object System

1 Motivation

We are living in times of ongoing rapid changes. New technologies support our everyday life in a way we could not even imagine a few years ago. New media are also disrupting our old fashioned way of thinking about education and pave the way for new visions [5].

Many different approaches for developing electronic teaching environments already exist. All of them have in common that they try to react on the changes taking place in our modern society. Acquired knowledge is becoming obsolete in a short amount of time and has to be updated and expanded. The term “Life-long Learning” is becoming more and more important. Teachers and students are often no longer present at the same location. They are not even active at the same time. Therefore lectures have to be available whenever and wherever one likes. On the other hand new technologies have to be made available as soon as possible. As an example consider the new WAP protocol which could make it possible to pursue a course using a mobile phone. This implies a highly

extensible and expandable system.

Teachers preparing a course often want to use already existing data. This data might be distributed among the network and might only be available in proprietary formats. It is an unbearable effort to collect and extract this data to feed it into a teaching environment. It has to be possible to easily include existing data with no further effort. It has to be possible to add or manipulatable information at any time without knowing any details of the underlying systems. When updated information has become available, documents have to be replaced. Nevertheless replaced information should not be lost as it shows the history of a course and might be interesting for some research. Thus a version control mechanism is desired as well.

Students need easy access to the teaching system, no matter which system they use (platform, browser, etc.). The way in which information is provided has to be adapted to the user's special needs and interests. The users' skills have to be taken into account as well as the capabilities of the system used. As an example, consider the network bandwidth with which the user is connected. Lower bandwidth could be considered by sending images of a lower resolution and just sound without the video. Users of different skills need information prepared differently, which means that different applications are provided for novice and expert users. Thus the system has to be highly customizable, not only by the user but also automatically by the system itself. Different views at the information space are desirable as well. As an example the information could be provided in different languages or with different, localized examples.

Another important point is the use of background libraries [7] to clarify terms. This includes dictionaries, encyclopedias and glossaries which again have to be adopted to the user's needs. As an example native English speakers would need an explanation of an unknown English term in English, whereas German native speakers would like to get a German explanation.

Though students may be distributed among the world they need the ability to discuss certain topics with the teacher or among themselves. Collaboration and communication tools are therefore required. This includes chat, discussion forums, video conferencing tools, etc.. Another important aspect is the possibility to add annotations and make personal notes to a topic which may only be seen by a specified group of others, maybe even excluding the teacher. This directly leads us to another interesting aspect in a teaching environment: user and group access management. Information presented in a course as well as additional information has to be protected from unauthorized access. The more sophisticated the user management is, the more configurable the system becomes.

Obviously the best user access management system is not worth anything if the transmission via the network is not secure at all. Highly sensible data has to be additionally encrypted. Imagine companies teaching their personnel internal knowledge.

For performance reasons hybrid systems are often desirable. Most of the courseware comes along on a CD-ROM. Only the logical part of a course could reside on a server. Thus lessons can be arbitrarily constructed using the information stored on the CD-ROM. Only additional or updated information has to be

fetches via the net. This keeps network traffic low without omitting special kinds of media such as videos. Hybrid systems are also kept up-to-date more easily, since outdated information stored on the CD-ROM can be replaced dynamically by new data from the server.

Linking is a good way to increase the value of information if used for defining the course flow [6]. This makes it easier to add or replace information used in a course. That way even statically stored information (e.g. on a CD-ROM) can be dynamically restructured to meet different needs. Since links are often subject to change or become invalid (especially links to external resources), a highly sophisticated link management is required to keep the system consistent.

Developing a teaching environment that meets all the functionality described above implies a big effort for implementing the base functionality. Most of this functionality has little to do with a certain teaching concept pursued. It requires a lot of time to develop the system so far that the actual vision can be put into practice. This paper describes how the cost for developing can be reduced dramatically when using a middleware system. This offers more time to concentrate on the crucial points of new ideas. In the following sections we present the abilities of the Dinopolis [3] middleware system which is highly integrative, expandable and configurable and provides an easy-to-use interface for building distributed applications.

2 A Distributed Environment

As already mentioned, distance learning gets more important every day. Teacher and students do not have to be physically present in a classroom. The teacher for example holds online lessons in his office, while the students follow the lessons from at home or from some Internet terminal on the campus. It is important to recognize that the students do not only need to follow the lessons conducted by the teacher, but also need the possibility to interact or communicate with the teacher during and after the lessons.

Please note that not only human beings are separated in such a distributed environment but also resources can be spread across the network. As an example, the lecture notes can be stored on the teacher's laptop while a chat tool may be located on the university server.

The main task of modern teaching environments is to bridge this physical separation of teachers, students and resources. This can be supported by using the Dinopolis middleware system. A middleware system adds another layer to the teaching environment. It decouples the network and data from the actual teaching application. It somehow resides between them. The Dinopolis middleware system allows the integration of arbitrary systems and provides a uniform way of access. Applications using Dinopolis do not have to worry about where the data is located and which protocol has to be spoken to retrieve the data.

From an application's point of view there is no difference if the information resides on a web server, on a database server or just on the local file system. The underlying systems may even be exchanged transparently. An important point is that integration within the Dinopolis system is not done on a common

denominator basis but on the contrary, additional functionality is added when needed. As an example a file system can be combined with a database system to allow the storage of meta data which the file-system itself does not support.

The most intuitive way to access data on a remote Dinopolis system is by “merging” the two Dinopolis systems, since it allows transparent access to remote Dinopolis instances. In other words the middleware layer bridges the physical separation and applications can be written as if they would run on one single system. Nevertheless the Dinopolis *External Access Gateway* concept allows access in various forms. At the moment this includes HTTP, FTP, WAP, LDAP, CORBA, RMI, etc.. CORBA and RMI are remote object systems which enable the use of objects over the network. In the case of CORBA [8] the Dinopolis object system can be used by programs written in various programming languages.

Thus no matter how distributed teachers, students and resources are, the Dinopolis middleware system acts as if they were all local. For a more sophisticated discussion of middleware systems and Dinopolis see [2].

3 Data And Course Flexibility

As already mentioned information is subject to change and the amount of information grows dramatically nowadays. Data must not be stored within a static teaching environment providing no possibility to adapt the data. It is also desirable to change the course flow dynamically to increase the value of information. As an example it has to be possible to add actual information to a course at any time. Thus developing a teaching environment built on static information is not worth the effort.

The Dinopolis middleware system allows to add, exchange and modify data at any time without needing to know any specific details about the underlying heterogenous information space. Version controlling guarantees that no information is lost. The distributed concept of Dinopolis allows to store data at arbitrary places transparently. Applications do not have to worry about that. This makes it relatively easy to develop so called hybrid systems, which combine local statically stored data (e.g. on a CD-ROM) with dynamically retrieved data (e.g. from the web). Due to the network bottleneck hybrid systems have the advantage that they do not have to omit bandwidth consuming information, such as videos, since they can be retrieved locally. Only updated information has to be downloaded via the net.

The course flow is best modeled using links. This makes it possible to insert or remove any kind of information without having to construct a complete new course. The problem with links is that they often become invalid or point to unintentionally changed data. Dinopolis comes along with a fully featured consistent link management system. Links do not point to a location but to an object. Thus even if objects are moved in the Dinopolis system, links remain valid. It is also possible to add meta data to links to provide additional information. Again the possibility to set links does not depend on the abilities of the underlying systems. As an example imagine a courseware CD-ROM. Since

a CD-ROM is read-only it is impossible to add or modify links directly on that medium. Nevertheless Dinopolis adds this functionality using an external link database (e.g. Oracle). This database can easily be exchanged transparently which then leads to a completely new course or course flow.

Dinopolis is able to handle arbitrary document formats. This is achieved through the Dinopolis internal document model, which follows the *Document Object Model* (DOM) specification of the W3C [9]. DOM is a highly accepted standard which supports the exchange of data between various kinds of different document formats. The internal document model again makes the whole system independent from the underlying ones.

4 An Extendable And Exchangeable Environment

A modern teaching environment not only has to be flexible concerning the course data and course flow. As applications evolve new technologies and requirements arise which require modifications of existing teaching environments.

Dinopolis is built on a completely modular basis. This allows to add or replace arbitrary parts of the system without producing unexpected side effects on the remaining parts of the system. Since Dinopolis is completely written in *Java*, it is possible to load new modules via the network even at runtime. Statically designed systems would not only have to be completely rewritten but also have to be redistributed among all users. As an example for a newly available technology consider the WAP protocol which could make it possible to pursue a course using a mobile phone. Dinopolis only requires a small WAP speaking module to be added and all applications using Dinopolis are becoming WAP aware. This means new technologies are available without any further effort. By the way adding new functionality to an application by hand would also require to be familiar with all the underlying details, which might be unnecessarily time consuming. Since Dinopolis is an *Open Source* project it is also guaranteed that modules supporting new technologies will be rapidly available.

Dinopolis is not only modular concerning external communication gateways. Internal parts of the Dinopolis system may be exchanged or added transparently as well to meet different needs. As an example let's take a look at the *User Access Management* and *Security* system which will be explained in section 6 more in detail. For some systems it is sufficient to define simple read and write access rights for users. Other systems require a much more sophisticated mechanism, such as a rule based one which only grants access if certain complex conditions are fulfilled. As an example students who have already passed an exam might get access to the results which are protected otherwise.

Teaching environments transmitting highly sensible data might also require to encrypt the information sent over the network, while a public system would not require it. This implies a *Security Manager* that can easily be adapted to different strategies. This also includes that some systems demand a user authentication based on smart-cards, finger prints or retina scans.

Dinopolis makes it easy to exchange or adapt internal parts to meet different requirements without the need to write a new application or modify existing ones. Dinopolis even allows to exchange the internal data structure or communication protocol used. At the moment data is stored according to the XML standard [1] and communication between Dinopolis instances is done using RMI or CORBA. As soon as new technologies and standards become available they will be integrated as well.

5 A Highly Customizable Environment

In a modern teaching environment we expect the main parts of the system to be highly customizable. The whole system has to allow users to turn on and off certain features according to their needs and their systems' capabilities. On the other hand applications and tools have to be customized depending on personal settings. Additional communication tools have to be provided to support a better information flow between users. The Dinopolis system is highly modular which allows adding and removing of integral parts, features, and services at runtime.

The users should have the possibility to choose between a variety of tools and applications that support their studies, depending on the users' skills and the used network connection.

The Dinopolis system makes it easy to write distributed applications. Existing tools can be reused and adapted for certain lectures as needed. It is easy to configure the system with different applications at runtime and the Java Classloader allows starting applications which reside on the local terminal, a CD-ROM, or to download them from an application server.

It is important that only those applications are part of a certain course, which are actually necessary.

Next it should be possible to customize the applications and tools themselves. As an example consider a video-conferencing tool which can be run with different frame-rates and resolutions. On the other hand it has to be possible to turn on and off special features like, for example, strong data encryption. Applications written for Dinopolis can make use of certain features or not, according to the environment in which they are used.

Apart from the applications and tools used, such a teaching environment has to allow customizing the users' view on the data stored in the information system. As an example consider a teacher who wants to adapt the data representation according to the different skills and needs of the users.

Therefore the Dinopolis system provides so-called Views, which support different data-representations. Using a highly sophisticated link management (see section 3) the data representation can be customized in a very high degree.

Another considerable point is that it has to be possible to decide where the data is actually stored. So it could be desired to store certain data on a file-system on the users' terminal or in a central database. It should be possible to transfer these data from one storage medium to another. Since the Dinopolis system

uses an internal data representation which is independent from the medium it is stored on, it is possible to store the data on different devices according to the actual configuration of the system.

6 User Access Management And Security

All distributed application have in common that their reliability and consistency heavily depends on the security and user access management. Unauthorized access to certain resources has to be prevented. This is also appropriate for a teaching environment. Just imagine students modifying and corrupting course data or exam results.

There exist many different approaches to solve this security task. Which one is appropriate for a certain environment depends on the desired degree of security. As already described in section 4 a simple security system differentiating between read and write access rights may be sufficient for some applications. More sophisticated environments could require a rule based access control system or secure (encrypted) transmission over the network.

One of the main parameters to measure well designed software is its level of reusability. In this case this means that security strategies have to be exchangeable easily. As already mentioned replacing the Dinopolis security concept is no complex task due to its modular structure.

Another point of high interest concerning access management in a teaching environment lies in the fact that data is distributed among heterogenous systems, all of which coming along with different or even no security management. It has to be guaranteed that users are forbidden to access data which they would not be allowed to access otherwise. Additionally if a system does not support any access control (e.g. MSDOS file-system) it has to be possible to add a security functionality. Dinopolis allows to integrate existing security mechanisms transparently. It is also possible to add access control to systems which do not support that by themselves. Thereto Dinopolis uses its integrated link management system. Access rights and rules are simply assigned to users and data through links which may be stored in any arbitrary external database. This concept makes it even possible to assign access rights to read-only systems (e.g. CD-ROM). Last but not least Dinopolis supports mandatory as well as discretionary access control. This means that access rights may be assigned to users as well as to data, which allows highly sophisticated combinations of rights.

For a more in detail discussion of access management in distributed systems and in Dinopolis see [4].

7 Conclusion

The Dinopolis open source middleware system presented in this paper is a powerful tool for building arbitrarily distributed teaching environments. It allows to integrate data from heterogenous information space transparently and makes them available through a common interface. Thereby functionality is not re-

duced to a minimum but on the contrary, additional functionality is appended where needed.

The modular concept of Dinopolis makes it easy to adapt the system to specific needs without producing unexpected side effects on the remaining parts of the system. Its high customizability makes it possible to configure the system to meet personnel needs. Additional communication tools such as chat, video conferencing, etc. may be used together with the teaching software without any further effort.

Courses may be constructed using all available distributed resources which also eases the creation of hybrid systems. Easy access to the system is possible not depending on the clients' platform or desired protocol.

The conglomerate of Dinopolis' features presented allows developers of teaching environments to concentrate solely on the crucial points of their ideas. Thus we believe that Dinopolis is going to play an important role in the future development of distributed teaching environments.

References

- [1] Bray T. et. al.: "Extensible Markup Language (XML) 1.0 (1998)", available online at <http://www.w3.org/TR/1998/REC-xml-19980210>.
- [2] Dallermassl, Ch.: "Aspects of Integration of Heterogenous Server Systems in Intranets - The Java Approach", IICM (Institute for Information Processing and Computer Supported New Media) (1999)
- [3] Dinopolis: "Homepage of the Dinopolis Open Source Project", available online at <http://www.dinopolis.org>
- [4] Haub, H.: "Aspects of Access Management in Heterogenous Distributed Object Systems", IICM (Institute for Information Processing and Computer Supported New Media) (1999)
- [5] Grudin, J.: "Groupware and Social Dynamics: Eight Challenges for Developers (1994)".
- [6] Van Dyke, J.R. & Sollins, K. R.: "Linking in a Global Information System, World Wide Web Journal - A Publication of the W3C" (1995), available online at <http://ana-www.lcs.mit.edu/anaweb/pdf-papers/tr-659.pdf>.
- [7] Marchionini, G. & Maurer, H.: "Digital Libraries as Components of Modern Computer Supported Learning Environments" in proceedings of EDMEDIA 95. World Conference on Educational Multimedia and Hypermedia, Graz Austria 1995.
- [8] Object Management Group, Inc.: "CORBA Language Specifications Summary", OMG (2000) available online at <http://www.omg.org/store/publications.html>.
- [9] W3C: "Document Object Model (DOM) Level 1 Specification" (1998), available online <http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001>