A Project Management module

by

Magnus Claesson, KTH

and

Jan Harada, UTH

Stockholm, December 2000

Examiner:
Prof. Seif Haridi
Teleinformatics, KTH

Supervisors:
Ass. prof Vladimir Vlassov
Teleinformatics, KTH
M. Sc. Carl-Axel Eriksson
Lecando AB

Abstract

A module intended to facilitate distributed information sharing on project basis have been designed and implemented for a Knowledge Management System (KMS) server application. The KMS server application is written for a Java ™ 2 Enterprise Edition (J2EE) server, using a three-tier application with a logical separation of the three tiers.

- A presentation tier where Java Server Pages and servlets are used to provide a dynamic and flexible user interface.
- A business logic tier where Enterprise JavaBeans (EJB) components are responsible for handling the business logic and delegating data storage transactions to the data tier.
- A data tier to store persistent data where other EJB components are responsible for the storage of data.

The module consists of basic functionality needed to administrate the project, a distributed file handler with support for a directory structure, file owner permissions, exclusive write locks and version handling. A newsreader is also incorporated in the module.

The objectives while developing this module have been to develop a scalable module that can perform well under high workloads, as well as providing a well-designed user interface. A small test tool have been developed in order to conduct load tests on the server application.

1 INTRODUCTION 1

1.1 Background 1
1.2 Purpose of this project 1
1.3 Structure of the report 2

2 OVERVIEW OF EXISTING TECHNOLOGIES FOR DISTRIBUTED APPLICATIONS 3

2.1 Architectures of web servers 4
2.2 The Java 2 Enterprise Edition 5
2.3 Software errors and testing 9
2.4 Java Server Pages 11

3 PROJECT MANAGEMENT SYSTEMS 15

3.1 Characteristics of project management systems 15
3.2 Overview of the market 15
3.3 Conclusions 18

4 ARCHITECTURE OF THE PROJECT MANAGEMENT MODULE 21

4.1 Design issues to be considered 21
4.2 Use Cases 23
4.3 Overview of the project management module 24
4.4 The project management module 25
4.5 The news reader 31
4.6 User interface 32
4.7 Web based Distributed Authoring and Versioning 33

5 IMPLEMENTATION 35

5.1 Design issues 35
5.2 Development issues 37
5.3 Classes and packages 39

6 LOAD TESTING 41

6.1 Tools 41
6.2 The test bed 43
6.3 Test strategy 43
6.4 Test results 44
6.5 Conclusions 49

7 CONCLUSIONS AND FUTURE WORK 51

7.1 Summary 51
7.2 Suggestions for future work 51

8 APPENDIX 53

8.1 References 53
8.2 Examples from the user interface 55

1 Introduction

1.1 Background

Lecando is a company that targets itself on the Knowledge Management System area of the market, and Lecando is developing products to facilitate administration and presentation of education components over intra- and Internet. Lecando's early technical platform was a flexible system, written entirely in Java and the architecture of the system was closer to the traditional two-tier application model than the three-tier model. Scalability of the system was substandard mainly due to two reasons; first that clustering of servers was not possible, and second that the relational database architecture was not optimized for performance. All dynamic presentation of content required calls to A relational database. The performance of a relational database constructed with conceptual schemas as a model, with no consideration with regard to performance is often poor.

Since the server application could not be clustered over different machines it lead to the situation that the possibility to improve performance via upgrading of hardware was extremely costly. To keep the competitive edge the technical platform have been rewritten entirely, using new technologies and developed with demands regarding scalability as well as all the inherited demands of the early system. This technical platform can be divided into modules where every module has a logically separable subset of responsibilities and functions. Examples of such modules are a chatserver with a couple of applets for the user interface or a course administration module.

Lecando wants to develop a module for project management for the new technical platform, the main objective of the module shall be to facilitate interproject communication and information sharing rather than project control and time management.

1.2 Purpose of this project

The purpose of this project is threefold.

- Evaluation of market: to examine a few typical and widespread project management tools on the market with regard to the availability of tools facilitating interproject communication and information sharing, choose some core functionality that must be implemented in the project management module and make suggestions on suitable functionality to be developed after the end of this master's project.
- Implementation: implement a project management module using the technologies that assure a scalable and expandable project management module that can work as a base to add new functionality to.
- Evaluation of module: to perform tests that will confirm or deny the scalability of the module under different workloads.

There are only few project management systems on the market that can be distributed in the meaning that they support administration of projects where the members can be geographically distant. Of these we have not been able to establish any project management systems that build on a technical platform that supports true distribution of the application components.

This is the true purpose of this master's project; to develop a project management module where the administration of the project as well as the underlying technical platform can be distributed.

1.3 Structure of the report

The remainder of this report is organized as follows.

Chapter 2 presents some of the existing technologies that are available for development of distributed application development.

Chapter 3 define two different types of project management systems, summarize key functions that are common for project management systems and present a few project management systems on the market.

Chapter 4 discuss design issues to be considered while developing distributed applications and presents an conceptual overview of the project management module we have developed

Chapter 5 discuss the implementation and issues related to the development process

Chapter 6 describe the load tests of the project management module that we have developed, presents the results of the load tests and the conclusions that we could draw from them.

Chapter 7 present our conclusions regarding reaching of our goals and how we got there. It also presents suggestion on how to improve our project management module.

2 Overview of existing technologies for distributed applications

A distributed system is a system where users invokes methods without being aware of that they reside on several machines. In a distributed system, the existence of autonomous computers can be transparent to the user. The user can type a command to run a program and it runs. It is up to the operating system to select the best processor, to find and transport all input files to that processor, and put the results in the appropriate place. In other words, the user of a distributed system is not aware there are multiple processors involved; to the user, it looks like one virtual uniprocessor. Allocation of jobs to processors and files to disk, movement of files between where they are stored and where they are needed, and other system functions must be automatic.

Developing and implementing distributed systems is not only time consuming but also complicated. It would be desirable to separate the methods handling the communication between the machines, and the business oriented methods, as well as objectifying the business methods in order to make them easy for a developer to understand, and implement. Object orientation provide means to make a complex problem concrete and easier to interpret.

Enterprises of today need quickly to develop and deploy custom application systems that are secure, flexible, scalable and reusable. It is imperative to reach data from remote locations, and the system need to provide reliability and concurrency without significant time loss. The demands has lead the development to distributed objects, which separate the business logic from the network issues, and allow objects on one machine to be used by client applications on different machines. This is accomplished by using a three-tier application, where the first tier handles the presentation, the second tier handles the business application; resources such as databases resides on the third tier.

The distributed object architecture is based on three parts: the stub, skeleton, and the object server. The stub and skeleton are responsible for making the object server, which resides on the middle tier, look as if it was running locally on the client machine. The object server residing on the middle tier is an instance of an object from the running machine. The skeleton is an extension of this instance to the stub, which resides on the client machine. The protocol responsible for this network communication is called a remote method invocation protocol (RMI protocol). When a client invokes a business method on the stub, the request is streamed through the RMI protocol to the skeleton, which in turn parses the stream in order to find what business method is requested. The skeleton then invokes the corresponding business method on the server. The value returned from the request by the server is then sent back by the skeleton to the stub. Finally the stub presents the value requested to the client application as if it had been processed locally.

The Java^{TM} 2 Platform, Enterprise Edition [9] fulfills these needs. It provides a single standard for server- and client-side multi-tier applications, where the client tier provides user interfaces, the middle tier provides client services as well as business logic, and the third tier provides persistent data management.

2.1 Architectures of web servers

We hereby use the definition of a web server as an application that provides access to web components that are pointed to by an Uniform Resource Locator (URL).

2.1.1 Static web pages

In the late 1980's the Internet increased its rate of expansion and on a conceptual level it was mostly used for file transfers. Client software was developed that could present textfiles directly which allowed the administrators of a server to put information regarding the files. In the early 1990's the technique to present graphical files in the browser was part of the growing interest in the internet. The browser still was almost only used for static file transfer. Soon the need for dynamic presentation became apparent, and different techniques evolved, including javascript and CGI-script mentioned below.

2.1.2 Two-tier architecture

From the need of persistence and context-dependent presentation CGI-script emerged. CGI-scripts made possible to distinguish two conceptually different mechanisms; presentation including logic operations for the presentation and storage with logic operations for storage.

This application model is called a two-tier architecture, where the first tier is presentation and the second is the storage. Often the storage is done by a relational database. A web server application handles a request from the client by invoking procedures on the web server which may affect the presentation.

A large number of web server applications of today use this two-tier architecture. However, as the size and complexity of web server applications have grown it has become apparent that this model has severe disadvantages. Even if the underlying technique is partly object-oriented this model spoils some of the benefits with object-orientation, such as hiding of data and delegation of methods to the data container. A required change in the storage tier often makes it necessary to make changes in the presentation layer which makes the maintenance and development of large systems cumbersome and error-prone. In many implementations the client's request forces the starting of a new server process to handle the request. This allow the server to handle requests concurrently, but decreases performance and scalability. Figure 2.1 [22] shows a schematic representation of the two-tier model.

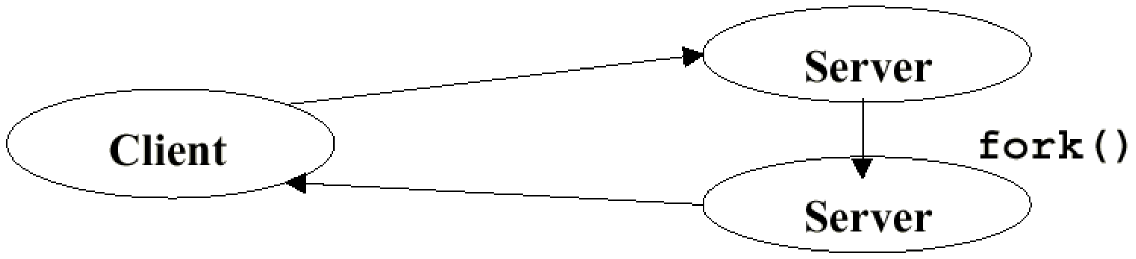


Figure 2.1 The two-tier model. A new server process is started to handle the request.

2.1.3 Three-tier architecture

In recent years an improved model have become widespread; the three-tier architecture **[5]**. It separates three conceptually different tiers and these are presentation, business logic and storage. Ideally this will make it possible to make changes in the one tier without being forced to make changes in other parts of the system.

A request is handled at the web server through invocation of the proper business logic, which in turn alters and/or retrieves data according to the request, sends them to the presentation layer where the data is presented via a response to the client.

This is not a perfect solution but offers more hiding of the methods operating on the data than the two-tier architecture. A schematic description of the three-tier model can be viewed in Figure 2.2 [23].

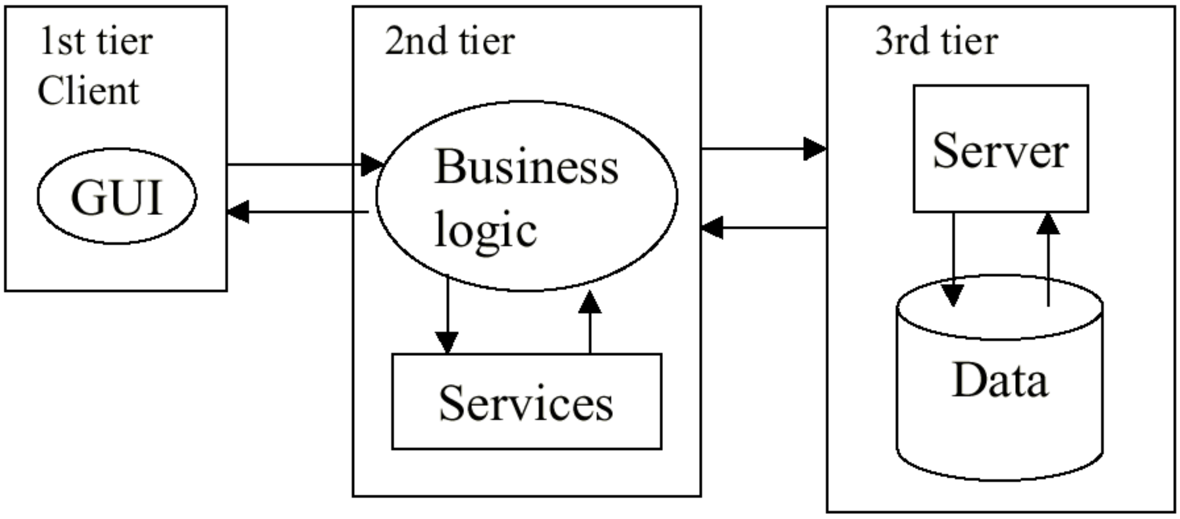


Figure 2.2 The three-tier model. (GUI is an acronym for Graphical User Interface).

2.2 The Java 2 Enterprise Edition

2.2.1 Overview

SUN has developed three specifications, the Java™ 2 Enterprise edition (J2EE), the Java™ 2 Standard Edition and the Java™ 2 Micro Edition. The Java™ 2 Micro Edition is intended mainly for devices with small processors, such as Personal Digital Assistants.

The Java™ 2 Standard Edition specifies the core classes, interfaces and packages of the Java language. J2EE is an extension of this specification, J2EE adds the functionality needed to develop and deploy a web server architecture, such as servlets and other webserver specific classes and interfaces. J2EE is the collection of a number of specifications, the core being Enterprise Java Beans [7] and there are a number of other specifications that the J2EE specification comprises of; among these the most fundamental are the Java Servlet Specification [10] , the Java Server Pages specification [11], the Java Naming Directory

Interface specification, J2EE Compatibility Test Suite, and the Java Message Service specification.

A J2EE server is an implementation of the J2EE specification. There are multiple benefits of using an J2EE server.

- Focus for developers: the J2EE application model separates levels of responsibility, the application developers can concentrate on implementing business logic and presentation, and the J2EE server takes care of the machine specific details.
- Portability: it is possible to write and compile the code once and then run it on different platforms with machine specific software. This includes the J2EE platform security model as well making it possible to maintain implemented security constraints across different platforms.
- Broad industry adoption: there are a number of third-party implementation of J2EE servers giving the application developers a better possibility to choose an implementation fitting their needs.
- Scalability: a J2EE server following the specification shall be able to distribute the components over different machines.

2.2.2 Enterprise JavaBeans, EJB

2.2.2.1 Overview

The Enterprise Java Beans specification is the core of the J2EE specification. As a part of the J2EE specification, it features the same benefits as those of the J2EE specification.

An EJB server allows the application developer to create and maintain persistent data in a distributed environment, the underlying management of data can be handled by a relational database management system, but it can be handled by other data management techniques.

Thus the EJB server allows the application developers to develop business logic, without having to indulge in system-level details of the complex issues of distributed and concurrent data handling. EJB components are entirely written in Java, and these components are named beans. There exists two main types of beans, entity beans and session beans.

2.2.2.2 Entity Beans

Entity beans represents persistent business data in an application, and can be seen as rows in a database. A good rule of thumb is that entity beans can be seen to model business concepts that can be expressed as nouns in a business application [6]. Entity beans describe both state and behavior, and allow developers to encapsulate data and rules associated with specific concepts. This makes handling of data associated with the concept secure and consistent. Entity beans can represent data in a database in several different ways. An entity bean can depending on its attributes span from a single row to an entire table.

Entity bean allows shared access from multiple clients. This is handled by the J2EE server's support of transaction management. Changes to an entity bean means changes to the database. This objectifying of data greatly simplifies tasks, such as changing state of a bean. Instead of writing SQL directly to the database, one can use a method through the bean. In addition, the objectifying of data also promotes software reuse, and simplifies development.

There are a few questions that should be confirmed in order to decide whether to model a business object as an entity bean or not.

- Representing persistent data – does the state of a business object needs persistent storage this suggests using entity beans?
- Providing concurrent access by multiple clients – does the business objects need to be available among multiple clients?
- Grouping together - do the business object represent a single logical record of data? This can be a single row or an entire table in a database.
- Do the business object need support for robust, persistent data management ?

There are two types of entity beans.

- Container-managed persistence beans have their persistence managed automatically by the EJB container. The container knows how a bean instance's fields are mapped in the database and takes care of all database methods automatically.
- Bean-managed persistence beans: they do all this work explicitly. The developer has to create all SQL code needed to manipulate the database. The EJB container informs the bean instance when it is safe to commit work, but provides no other help. The bean instance conducts the persistence work itself.

2.2.2.3 Session Beans

Session beans are used to implement the business object that represents business logic. The state of such an object initializes on behalf of a single client, and can therefore not be shared among other clients. A sesssion bean is a logical extension of the client program running, and contains information specific to one user. In contrast to entity beans, session beans does not directly represent shared data in a database, but instead define methods to manipulate data.

They are components that allow clients to perform tasks without being concerned with the details that make up the tasks, which allows developers to manipulate the bean without impacting on the client program code. There are two kinds of session beans, stateful and stateless session beans.

Stateful sessions beans do not directly represent data in a persistent database, but they can access and manipulate data on behalf of a client. They are designed to maintain a conversational state on behalf of a client, therefore business-oriented logic should be modeled as stateful session beans. Since they represent non-persisting object they are destroyed when the client's session is over – or when the server suffers a crash.

Stateless session beans are designed only to provide server-side behavior. They contain no user-specific data, but instead the EJB architecture provide ways for a single stateless session bean to provide services for multiple clients. It is a business object that provide generic services to multiple clients. Such an object does not need to maintain any client specific state information, so the same bean instance can be reused to service other clients. Since it is reusable and not tied to a specific client it can requires fewer system resources which may enhance scalability of the application.

2.2.2.4 Implementation of beans

In order to implement an enterprise bean, one is obligated to define two interfaces and at least one class: the home interface, the remote interface, and the bean class, and/or a primary key class, depending on whether the bean is an entity bean or not.

The remote interface defines the methods available for a client interacting with an enterprise bean. For every method in the remote interface, there is a corresponding method in the bean.

In other words, it defines the set of business methods available to the clients. This interface allow the client to perform the following methods on a reference to an enterprise bean instance:

- Obtain the home interface.
- Remove the enterprise bean instance.
- Obtain a handle to the enterprise bean instance.
- Obtain an entity bean instance's primary key.

All these methods are business oriented in the way that they all represent an action, as naming, setting, and getting.

The enterprise bean's home interface defines the methods for the client to create, remove and find EJB objects of the same type. This interface allow clients to do the following.

- Create a new enterprise bean instance.
- Remove an enterprise bean instance.
- Retrieve meta-data for the enterprise bean through the interface. The interface is provided to allow application assembly tools to discover the meta-data information about the enterprise bean at deployment time.
- Obtain a handle to the home interface, which provides the mechanism for entity beans. The home interface of an entity bean provides methods for finding existing entity beans instances within the home. A client that knows the primary key of an entity object, can obtain a reference to the entity bean by invoking a finding method on the entity bean's home interface.

The primary key provides a pointer to the database, and therefore only entity beans need a primary key.

These classes provide methods for creating, removing, finding, and using a bean. The last class needed for a implementation is the bean class. The bean class implements the bean's business methods. However the bean class does not implement the bean's remote- or home-interfaces, but it does need methods matching the signatures of the methods defined in the remote interface. It share responsibility for two specialized categories of methods with the remote interface: the create- and finder- methods. Furthermore it must have methods corresponding to some of the methods in the home interface.

The bean object resides in the middle tier, and handles the business methods requested by the client, through its home- and remote- interface. The container surrounding the bean is responsible for interactions with the server. A schematic presentation is shown in Figure 2.3 [23].

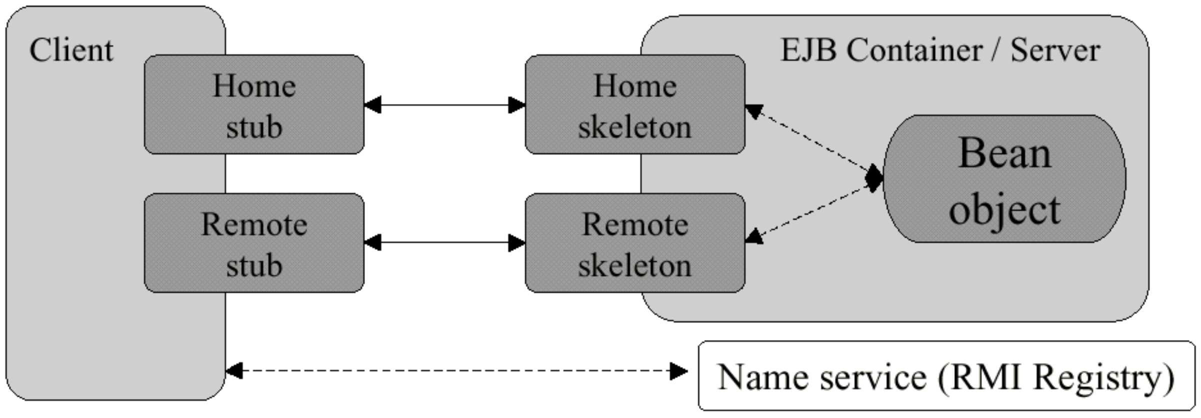


Figure 2.3 Description of interaction between a client and the server

2.3 Software errors and testing

2.3.1 Problems with software development and testing

When developing software it is important to have the ability to deliver stable and error-free products. In general the complexity of a system increases with increased functionality and this makes it difficult to make a reasonably correct estimate of time and resources that is needed to develop the product.

The traditional way of developing a software system has been to do a thorough specification and modelling, and then it is up to the development team to implement this specification. As timeschedules often are tight, testing can be sketchy at best on a component level.

In the end phase of a development process when all of the components have been assembled, the system is tested and often this testing is done by people separated from the development team.

As there is a vast number of possible states in a software system, and commonly a large number of possible test cases as well, high quality of the end product – where high quality is defined as a low number of errors – normally cannot be guaranteed. Instead quality can be viewed as probability function of the tests, where an increased number of tests where errors have been found increases the probable quality. Since it is not feasible to test the entire system the tests have to be selective, and test cases are often constructed based on typical user scenarios. Empirical experience suggests that there normally is a linear correlation between size of an application – where size is measured in lines of code of the application – and the number of software errors have implemented while creating the application. This has lead to the somewhat pragmatic approach of some software development companies, that they will not release a commercial product until a certain number of precalculated errors have been found during the test phase. However, the correction of found errors can be a tedious task in a large and complex system, on a distributed system it can be virtually impossible to recreate the state that triggered the error. It can also be difficult identifying which components of the system that are responsible for the error. Often the programmers have forgotten details about their work, and do not have the overview of the system to be sufficiently certain how a error correction will affect the integrated system, and usually the time-schedule is even more hard pressed. Ad-hoc error corrections often introduces new errors which makes another test phase necessary, and so forth where the number of errors hopefully diminish exponentially.

Often programmers get involved into new projects which produce error reports and their fixing an unwelcome increase in workload. During these circumstances the risk increases for a lowering in quality of the work, and documentation of the system and its components often is neglected.

Since software development progress towards projects in the magnitude of large-scale industry projects, large efforts have been made world-wide to address the problem of errors in software. It is quite easy to construct methods and software for component testing and quite complex to construct methods and software for application testing. Therefore component testing is more widespread than the latter and there exists numerous ways to do it, from ad-hoc to systematic.

2.3.2 JUnit – component testing

In Java errors normally are more infrequent than in other languages, much thanks to Java's systematic exception handling, clever compiling warnings and Java's total lack of programmer handled memory pointers. However, errors are always unwelcome and JUnit [12] is a systematic way to decrease them.

JUnit allows the developer to set up a test class for every class in the application, and then construct tests on method level in the test classes. JUnit then supplies the framework needed to automate the process of running the tests at compiletime.

How to use JUnit can briefly be summarized in the following steps:

- First create classes and their methods.
- For every class that is constructed a similarily named test class is created.

In every test class the testing consists of three stages:

- Instantiating of the class.
- Calling methods with values for which the output is known.
- Comparing actual results with the known output.

These tests should be run after compilation, preferably as the last line in a compilation makefile. This way, at compiletime an OK or an error message is displayed.

JUnit also facilitates refactoring of code, ie when you want to optimize or rewrite your code you can rely somewhat on the tests assuring you that your methods work as intended.

Since only a minor amount of time has to be sacrificed in order to use JUnit there is no reason not to use them, or a similar test procedure. But it is important to point out a few key issues regarding the use of JUnit. First it requires imaginative test construction in order for the tests to be of any use at all. Second, whenever changes to the API has to be done, changes to the tests must be done accordingly. Third and most important, no tests of the user interface is possible. You can test interaction between several components if you have written service classes that use several components, and this is a major plus, but still many errors can be implemented during the construction of the user interface. These errors – although usually trivial to correct - imply low quality to the end user.

JUnit takes a little extra development time to incorporate in the development process, but we strongly recommend the use of any systematic component testing tool, such as JUnit, in order to improve software development quality.

2.4 Java Server Pages

2.4.1 History of dynamic presentation on the web

In the early days of the Internet only static contents could be presented on the web. The need to create dynamic content presentation pushed the development forward. Common Gateway Interface, CGI, was the first attempt. It provides the ability to retrieve non-static data for a client request, managed by an application that runs on the server. CGI-scripts can be written in a multitude of languages supported by the server. CGI-script was used for a diversity of functions, such as hit counts on web pages. Performance was poor and maintenance of state was problematic. With the creation of Java in the mid 1990s browsers started to support Javascript [2]. Javascripts are small interpreted programs in a language that has similarities to Java, and Javascripts are embedded in the HTML [1] pages. The ability to download and to run bytecompiled Java programs with strict security restrictions also became widespread and a norm for browsers. Poor performance, stealable code and the slight awkward procedures in handling these modules created a demand for new solutions.

From the early versions of the SUN's Java implementations servlets are supported, and it is simply put compiled Java programs running on the server, carrying out tasks that is related to the clients requests. Here performance is also an issue – not more than a few thousands of Java threads can be created on even powerful servers of today, and in some implementations a new thread is created for every client. This puts a limit on the number of clients that can log on to the server systems that are built with Java servlets.

More powerful, truly distributed languages have been developed as an alternative to Java, a good example is the Oz language [14]. Oz supports extremely lightweight threads and creates a thread at 1/60th of the time to create a thread in SUN's implementation of Java. It is also possible to create over 100.000 threads on a standard desktop PC. Unfortunately the market does not always support the best solutions but often the most widespread one, and the dominance of Java in server solutions today is almost total.

2.4.2 The Java Server Pages approach

Servlets are compiled Java code that is running on the server, returning created HTML pages as response to requests. When a change is done to the servlet it has to be recompiled, put in the correct package and the servlet engine has to be restarted, often a tedious task.

Java Server Pages is a specification, and a JSP page can conceptually be visualised as specification or a template for a servlet. JSP can use the full functionality of the Java language, as it is compiled to a servlet during runtime. It can instantiate EJB components, access them and present data from them to a browser. JSP gets converted into Java Byte Code and is essentially a interpreted language. JSP uses Inner Classes or Declarations for separation of code. The JSP page accesses the web server through dynamic linking and there is no need to write the request to a pipe or file, starting a program on the server and reading back the output. The state can easily be kept by the Servlet host.

The idea is to embed Java code in the HTML pages instead of embedding HTML in servlets. When a fresh JSP page is requested the JSP page compiler generates a servlet, the servlet is then run on the client computer to present HTML code in the browser. Requests for data is taken care of by direct access to other components such as EJB.

With the SUN's reference implementation a JSP server is supplied that takes care of the creation of servlets based on the corresponding JSP files.

The order of processing requests and responses when using JSP is briefly put: a client makes a request for a JSP file to the JSP web server, see Figure 4.3, the JSP web server forwards the request to the JSP file, the JSP file typically retrieves data from other components such as EJB components or servlets and then sends back a response object to the JSP web server which in turn sends the processed JSP page as response to the client. The JSP specification does not allow posting to a JSP page even though some JSP server supports this, in order to keep an application portable, servlets must be created to handle posted data.

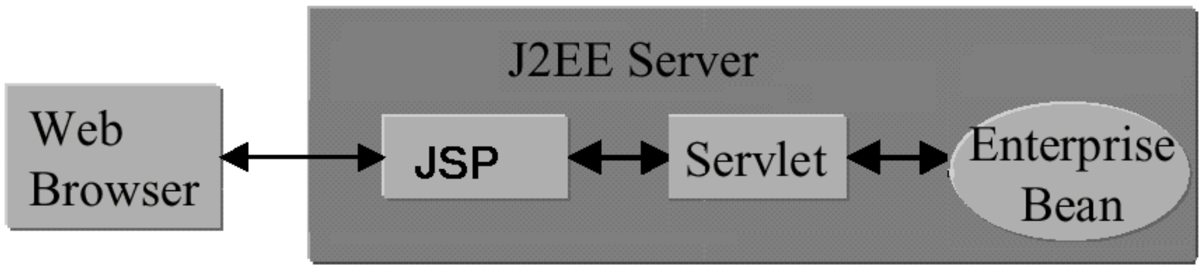


Figure 4.3 Description of a request for a JSP file

2.4.3 Java Server Pages tags and syntax

Unlike HTML JSP tags are case sensitive, here are a few of the most common tags, assuming that Java is used as the scripting language.

```
<%@ page import="javax.servlet.HttpServlet" %>
```

Page directives apply to the entire JSP source file, it can among others be commentary or for specification of imported packages

```
<% include file="test.gif" %>
```

The include directive embeds the contents of another file in the main JSP file

```
<%=printThis( ) %>
```

The “=” is a shorthand for the Java “out.println” and the evaluated expression is cast to a string and streamed to the browser

```
<% // any valid Java code %>
```

Declarations, expressions and any other valid Java code, as can be expected declaration must preceed use of the declarated object.

Code embedded in a JSP page will be compiled and read by the J2EE server, neglecting the HTML code structure. This can be considered a bit awkward at first but works in a simplistic and straightforward fashion. When using conditionals in a JSP file, the output of HTML will be controlled by the conditionals, as can be seen in the following example:

```
<HTML>
<BODY>
<%
           final  int alwaysPositive=1
           if(alwaysPositive>0)
           {
%>
                      <H1>This will always be displayed</H1>
<%
           }
           else {
%>
                      <H1>This will never be displayed</H1>
<%
            }
 %>
</BODY>
</HTML>
```

Even if these lines of code have been indented to ease reading, it is easy to understand that the code in JSP is not user-friendly to read or create. When the amount of code increases it becomes hard to get an overview of the code since it is intervened with the HTML tags, and no good texteditor modules are available to help the developer with indentation or syntax checking for JSP pages.

Since the JSP page is compiled to a servlet during runtime, it is not possible to detect standard compile-time errors in a JSP page during compiling and deployment of an application. Thus JSP syntactic errors are detected runtime and this makes JSP pages a bit awkward to use, on a semi-large application this deployment may take fifteen minutes with the standard java compiler from SUN. Great care must be taken to check the JSP pages before deployment to avoid vasting valuable development time due to minor spelling errors.

It is important to note that JSP pages is actually a shorthand for servlets, it is not essentially a way to create HTML pages with some dynamic presentation. Instead of writing a servlet that can stream HTML to a client one writes a HTML page with embedded Java code that will be compiled into a servlet runtime when the JSP page is first accessed. However moot this point might seem, it may serve as a comfort to the developer that it normally is faster to create and maintain JSP pages than it is to use Java servlets.

One should not confuse JSP with JavaScript, JavaScript is interpreted in the browser and JSP pages are servlets that return HTML and possibly JavaScript.

The JSP specification from SUN claims that one of the advantages with JSP pages is that they can be altered without having to stop the server application and recompile it, only the altered JSP pages are recompiled during runtime. This is not necessarily so in all J2EE servers. Since the JSP pages are compiled at runtime it is not possible to detect any errors during deployment of the application, if the compiler is not typechecking JSP pages explicitly. If it does not, it can be a timeconsuming task to work with JSP pages.

3 Project management systems

3.1 Characteristics of project management systems

There is a great diversity regarding functionality among the project management systems since they are aimed at different parts of the market. However, it is possible to distinguish and characterize some key concepts that are implemented in many project management systems. Among these key concepts are access restriction of project information, the possibility to share and distribute files, tools for handling what has to be done and what has been done by whom, calendar function with possibility to book project meetings and so forth, a messaging system for project specific messages, resource management tools such as time reporting and budget monitoring tools.

In spite of the similarities of project management systems, it is possible to distinguish two types; one that aims for facilitation of project control and the other that focus on information sharing.

3.1.1 Project management systems with focus on control of resources

This type of project management system give the user possibility to control and monitor the resources of the project, typically the user is the project manager with responsibility to balance the work amongst the resources available and to be able to keep track of costs, calculate time consumption of tasks and schedules for the individual resources and for the project as a whole. This is the most commonly used commercial type of project management tools, with a clear objective to facilitate monitoring of the economic factors of a project.

Little is done to facilitate the information sharing and the actual distributed work of the project and as we believe this is either disregarded or facilitated by other system components.

3.1.2 Project management systems with focus on information sharing

Twenty years ago, it was hard to believe that a number of volunteers could make a competitive threat to multinational companies investing large sums of money into research and development. With the growth of the Internet, a new type of software development has evolved. Unpaid part-time developers making a synergistic effort to produce software that can match those of even the largest of software companies, such as Microsoft and SUN. Existing management systems are the result of this new type of software development.

3.2 Overview of the market

There are numerous applications that can be used to enhance the control and administration of projects. There are only a few project management systems that focus on information sharing, the reason for this might be that there is not a great need for such a product. Possibly it lies in the unproved hypothesis that for being commercially successful, a project should not endorse personal initiatives regarding the time planning and delegation, but instead centralize the control and administration.

3.2.1 Project management systems with focus on control of resources

To get insight in how different project management systems of this kind meet demands of today, the following project management systems were examined:

- Microsoft Project [18]
- Microsoft Outlook [19]

On the market of today there is a number of systems developed for the facilitation of project management, many of them aimed for users involved in large and serious projects where it is important to maintain and share much information about the project's status. In this type of projects, the project managers can afford to spend time learning to use a complex system and spend time assembling and sharing information regarding the project, since the alternative might be an ill-run project wasting valuable resources.

The applications we have examined are commercial and with different objectives, intended for different parts of the market. All efforts to manage projects comes with a cost – the overhead for administration of tasks and users is the price that has to be paid regardless of how it is performed. To be able to perform this master's project in a professional manner we plan to use a project management system to keep the project on track, and to get a chance to evaluate and learn as much as possible from this project. At first we were concerned that the overhead would not be worth the benefits but now we have chosen to use Microsoft Project [18] for this task. And we believe that as a bonus we will get a good idea of the end-user's need and demands of a project management system.

3.2.1.1 Microsoft Project

Microsoft Project by Microsoft is a project management system which facilitates for project managers, with emphasis on timeplanning, delegation of tasks and resource management. The task management functions are powerful and calculates the completion dates of sequential and parallel tasks based on delegation and completion criteria specified. When creating tasks and linking them, ie task A has to be completed before task B, the project manager can specify criteria for the task such as "no later than". Tasks can have duration times and can be assigned to resources. Resources are identified by name and have work schedules, they can be assigned cost per hour for the assigning. A resource is typically a human but can be anything, such as servers or rented equipment.

The project managers can easily monitor workload on a resource and what tasks are assigned for the resource every day of the project. The project manager specifies an end date or a start date, assigns tasks for the resources and during assigning the time plan will be created dynamically, either showing what date that the project has to start on or which date that it has to be finished. It is not meant to specify a start date and an end date and then use the application to find out how hard the project must be driven to be completed on time. In project management systems the term "milestone" is often used. It represents the completion of a phase in the project and in Microsoft Projects milestones can be incorporated as tasks. The task list is expandable in the hierarchies so the project manager can view only the relevant tasks for the current planning. The cost for tasks and the project is updated dynamically so it is possible to track costs continually during the project.

It is possible to make report printouts of the numerous graphs and charts and one can include Office component documents in the graphs. We believe that this product is a tool for facilitating the running of projects with a strict hierarchy; there are numerous functions for the project manager to plan, track and delegate time and resources. It is possible to publish chosen parts of the project on the web and a email function can be used for interchanging status or project information. This is an application and not a distributed system aimed to facilitate information sharing between all the members of the project, there is no support for such things as authentication of members or to give them access to edit parts of the project planning. Nonetheless we believe that it is a versatile tool for small to medium sized projects where the project manager wants to have total control of the administration of the project.

3.2.1.2 Microsoft Outlook

Microsoft Outlook is another product in the Office family by Microsoft. All Office products are integrated to a certain extent. They share much of the user interface and base functionality. This is an advantage if one is used to and content with their user interface and otherwise a it might be a certain learning threshold for those who does not want the automation of services that one is used to have control of. Having said that, Outlook is an application that can handle emails and that is probably how most users use it. It does not actually qualify as a project management system as it is mainly intended for single users who has a need for managing meetings, keeping track of contacts and creating non-hierarchichal tasks.

But it supports intranet usage and then it is possible to share calendars and book meetings with other users and then it, as we believe, can be of great help during projects to facilitate intraproject information. Used in this way it is product demanding discipline from its users to avoid chaos, but being simplistic and intended for non-power users perhaps this is not the right product to be used for all potential small and loosely handed projects.

There have been several grave security flaws on Microsoft products, most of them have been related to the Visual Basic scripting support for emails and documents. This support can be turned off to increase security and Microsoft always supplies security fixes with great speed. Nonetheless, for a serious, expensive or sensitive project the security issue have to be considered regardless of which Microsoft product is being used. Microsoft have a long tradition of making software intended for standalone computers and came late into the competition for Internet products, perhaps related to low knowledge of web based issues among their employees. They have not, in our opinion, addressed the security flaws to their full extent for strict commercial reasons. With or without errors in their products Microsoft can sell vast amounts of software due to their position on the market, and it costs money to find potential security breaches.

3.2.2 Project management systems with focus on information sharing

There are not many implementations of this type of project management systems We have examined one implementation.

- Projectplace.com [20]

3.2.2.1 Projectplace.com

Projectplace.com is a web based client-server project management system developed by the swedish company Projectplace International AB. Projectplace.com allow project members connecting to a server via a browser, sharing project information and administrating it. The clients pay for a file space on the server, this file space is shared among the members of the project. Each member of a project is authenticated by a login and password. All the data stored is encrypted and all the interaction with the client is encrypted with the HTTPS SSL encryption, if the client software supports this.

Projectplace provides a file manager where files and their meta-data can be stored on the webserver. This file manager also supports version handling, where an old version can be retrieved from the webserver. Another feature is template documents, ie base documents with correct format and information to begin with. This feature seems a bit redundant as it can easily be incorporated in the project via agreements, but nonetheless the feature exists.

Users of the Projectplace.com system can use a standard browser without any plugins and this, sets limits to the user interface. The file handling logic can be improved if the client accepts and downloads a file transfer applet that allows the client to modify the files without saving them locally, the files have to be downloaded to the client computer nevertheless.

3.3 Conclusions

There are quite a few project management systems available on market today, and most of them do not provide distributed project management, ie allowing seamless geographic separation of clients.

The base functionality is quite similar between the different types of project management systems, regardless of implementation. Among the most common are facilitation of distributed messaging, similar to emails and newsgroups, task lists and calendars, schedule of meetings, administration of project resources and control of economy and cost.

But most project management systems are focused on aiding the manager of the project. If there are more members within the project, they can be viewed as resources that can be used and controlled during the project span.

Enterprises of today often separate their workforce in projects, this modularization of the workforce is useful when outside consultants are being hired for specific purposes, since the employees are used to work in different project teams. Considering this it is a bit surprising that truly distributed project management systems, that focus on decentralization and information sharing are not more common.

With regard to the aim of enhancing the existing system of Lecando by a module facilitating interproject communication and information sharing, existing systems for project management have been evaluated. Microsoft Outlook and Microsoft Project do not provide any possibility to manage distributed projects in an acceptable fashion as they are not aimed for this purpose. Both of these systems provide limited support for intranet usage and thereby requiring the project to be administrated on a local workstation or within an intranet. Projectplace.com allows a project to be distributed, but cannot fulfill the requirements of Lecando. First, it is not a module that can be incorporated into the existing system of

Lecando. The internal handling of data structures and security can not seamlessly be handled in Projectplace.com. Second, to our knowledge it is not built on a technical platform that fulfill the demands regarding performance and scalability that is specified by Lecando.

Therefore the decision was made to develop the project management module described in the following chapters.

4 Architecture of the project management module

This chapter presents the conceptual architecture of the project manager module that we have developed and implemented. We define a module as an application component that is depending on other application components in order to work. Our project management module for example, is relying on other components to take care of user data creation, storage and modification.

There are several design issues to be considered for development of a distributed application, the key issue being distribution. The project management module we have developed, hereafter referred to as the PM module, can be described to be distributed in two ways. First, it allow the members of a project created in the PM module to share information in a distributed environment, ie the project members can be geographically separated and still share information in a structured and controlled environment. Second, the underlying technology allow the project data and software components to be distributed over more than one processor and machine.

The PM module provides three conceptually separated parts.

- The core administration of a project such as adding and removing of members, controlling project specific attributes, handing over responsibilities of a project, as described in chapter 4.4.
- The file manager, a file system where project members can upload and share files in a directory structure, lock files to prevent deletion and access older versions of uploaded files, as described in chapter 4.4.1.
- The news reader where members of the project can post messages with or without file attachments. These messages can only be read and replied to by other project members. The news reader is described in chapter 4.5.

4.1 Design issues to be considered

4.1.1 Availability

One of the key aspects of a web server of today is its availability since more and more of the web servers of today perform business critical operations, and interruptions in these operations will have a severe effect on the enterprise.

Availability can be defined and measured in different ways, such as percentage of the time a web server can respond to requests or number of server breakdowns per some time unit. There are however a chain of components that must work in order for a client to be able to receive a response from a web server, eg the clients ISP, the web servers ISP, the web servers LAN, often a multitude of software components on the web server. For most enterprises, little can be done about the external factors and many ISP of today have no guarantees of QoS regarding their availability.

4.1.2 Fault tolerance

Much can be said concerning the concept of fault tolerance. Fault tolerance in general can regard different abilities of a system; it can be that the ability to identify and disregard corrupt input or to handle different faults in the system in some appropriate manner, eg. to save non corrupted data so that the state of the system can be restored after a crash recovery.

In a distributed system fault tolerance is regarded on a non-centralized basis, ie the ability for different parts of the distributed system to be able to perform some functions even if another part of the system has ceased to function. Distributed fault tolerance is difficult to implement and almost always use time-outs to detect crashed parts of a system. These time-outs have to be tune to avoid accidental restarts of functioning parts, ie system components with an increased response times due to high workload.

4.1.3 Concurrency

Concurrency means the ability for a system to handle multiple users in a correct fashion, ie the ability to allow simultanous modification of shared data in a manner so that the altered state of the system reflects a state which is possible if the modifications were made sequentially. Updates of data is often referred to as transactions, and correct transaction can be offered by almost every implementation of a relational database system, but the performance drawbacks between the different models of transaction management that are implemented differs quite a lot.

4.1.4 Scalability and performance

Web server performance can be measured in a variety of ways, one simple definition of web server performance is the average response time, where low numbers indicate high performance. Scalability is the ability to increase load factors, such as database size and number of requests, without severely decreasing performance. Scalability can also be the possibility to keep performance via plugging in more hardware in a parallel fashion, ie not via purchase of super computers but rather via installing more processors of similar type.

For a web server application the performance has increased in importance. When broadband Internet services have become an abundant resource, at least in Sweden at the end of year 2000, tolerance for long wait times have decreased among the Internet community. Web servers where there is some sort of session state maintained, eg a login, and where there is data that should be accessed and modifiable by the users, often suffer from bad performance.

There seem to be little understanding regarding the fundamental differences of application software and distributed system software, and the difficulties of maintaining a distributed state in a multi-user environment, and the performance drawbacks it brings along is not common knowledge.

One of the biggest bottlenecks of stateful distributed systems is the underlying database. The conceptual schemas that computer science students are taught at database courses are simple to grasp and construct, and fitting for applications with a small number of users. For multi-user, distributed systems the underlying data handling must be carefully planned, where performance optimization and reducing of data handling overhead must be considered in detail.

4.1.5 Security

Computer system security is almost always a compromise between several issues such as user-friendliness, performance and cost [4]. Often when security is mentioned cryptography is the topic, but cryptography should only be used to enhance the security of a system, it cannot be the only implementation of security for a system to be regarded as safe. What also must be kept in mind is the concept of security as a chain of links, where a weak link will jeopardize the entire chain no matter how strong the other links are. Many of the security breeches and leaks are made by employees that are trusted with passwords etc. A weakness that often is exploited by attackers are wrongful setup and installation of security components, such as the operating systems.

Security has a tradition of being overlooked in the Internet industry, and even if there exists security methods and systems that will make brute-force attacks more or less computationally infeasible, installing of such systems still is no guarantee for having a secure system.

4.1.6 Session

When a client interacts with a server, when the server needs information from the client, there are two general ways to handle this.

- The client can send all requested information every time and the server maintains no client-specific information
- The server can hold the client-specific data that is consistent during request or client-specific data that requires user affirmation to change.

The latter of these two ways is called a session. Sessions might have an impact on server performance, positive or negative, depending on several factors such as request workload, number of sessions kept, potential bottlenecks in the system. Generally speaking, there might be a gain in performance using client sessions if the client-specific data the server requires takes much time for the server to get hold of. Often sessions are kept in the servers RAM memory and is therefore is quick to access, but this memory often is a precious resource and can be run out of if not kept under surveillance.

4.2 Use Cases

4.2.1 Actors and Use Cases

We have used the UML [3] Use Case model to define our functional demands on the module, for readers not familiar with the object-oriented modelling of UML a brief introduction to Use Cases might be appropriate.

Users of a system are referred to as Actors, modelled visualized in the model by a simple line picture of a human. Actors are external entities – people or other systems - who interact with the system to perform a conceptual operation. This interaction is called a Use Case, visualized in the model by an oval with text describing the interaction.

Use Cases are a simple way of defining what operations a system shall be able to perform, and it is easy to grasp for non-developers. The problem with Use Cases is to decide on what level of detail the Use Cases shall be constructed. It seems that the general agreement is to keep Use Cases on a conceptual level and not create hundreds of Use Cases for low level functions.

4.3 Overview of the project management module

The implementation is that of a J2EE application, it incorporates the standard request handling, transaction management, response handling of a J2EE enterprise application. The dynamic presentation of the response is partially done in the JSP pages. Unfortunately some of the user-interaction had to be done using javascript, and this is an effect of being forced to use the semi-static client that a browser represents. Figure 4.1 shows a schematic representation of the three-tier architecture.

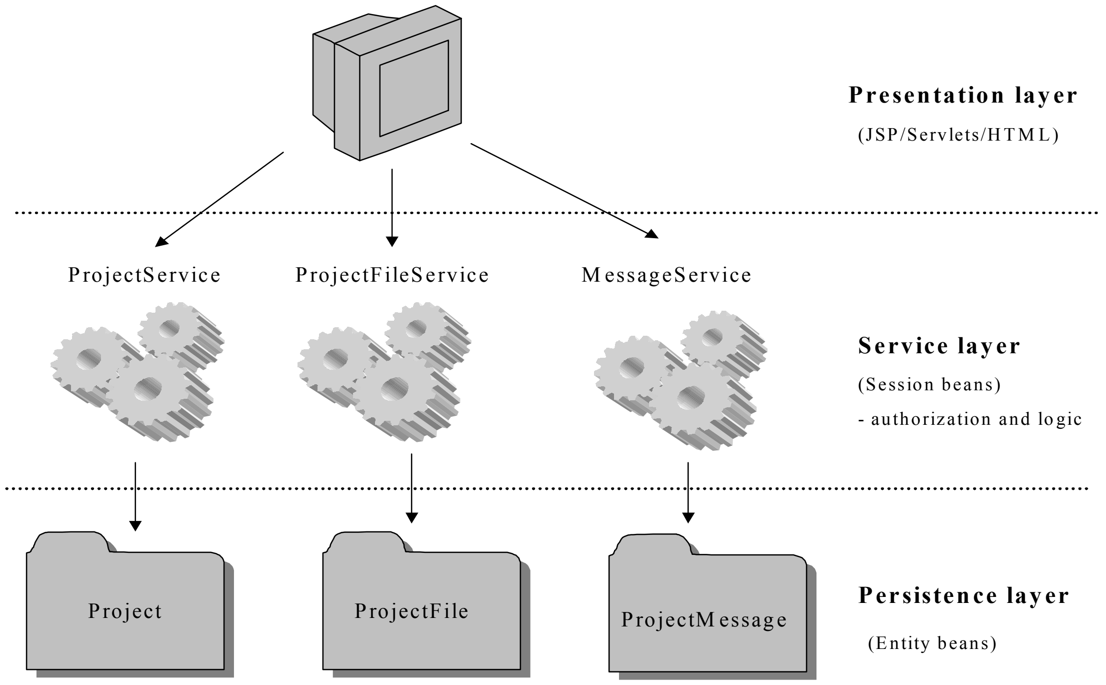


Figure 4.1 Overview of the three-tier architecture

As mentioned in the beginning of this chapter, the project manager module has three distinct parts.

- A representation of the project, which is the container of data related to the project such as memberlist etc.
- The file manager, which maintains the directory structure and files for the project.

- The news reader where members can read and post messages.

In Figure 4.2 an UML [3] diagram of the interaction within and between these three parts are shown.

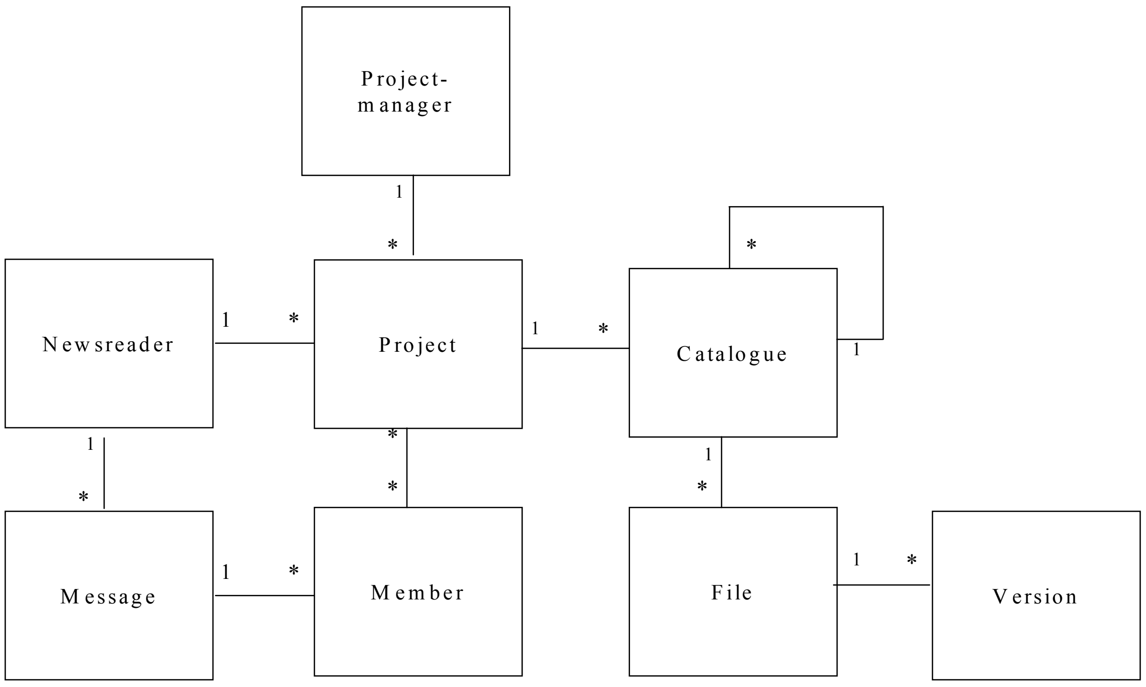


Figure 4.2 Conceptual UML class diagram of the project manager module

4.4 The project management module

The conceptual functionality of the project module was formed with the help of Use Cases. The Use Cases we constructed for the entire project module is shown in figure 4.3 and Table 4.1. More fine-grained Use Cases are presented for the file manager in chapter 4.4.1 and the for the news reader in chapter 4.5.

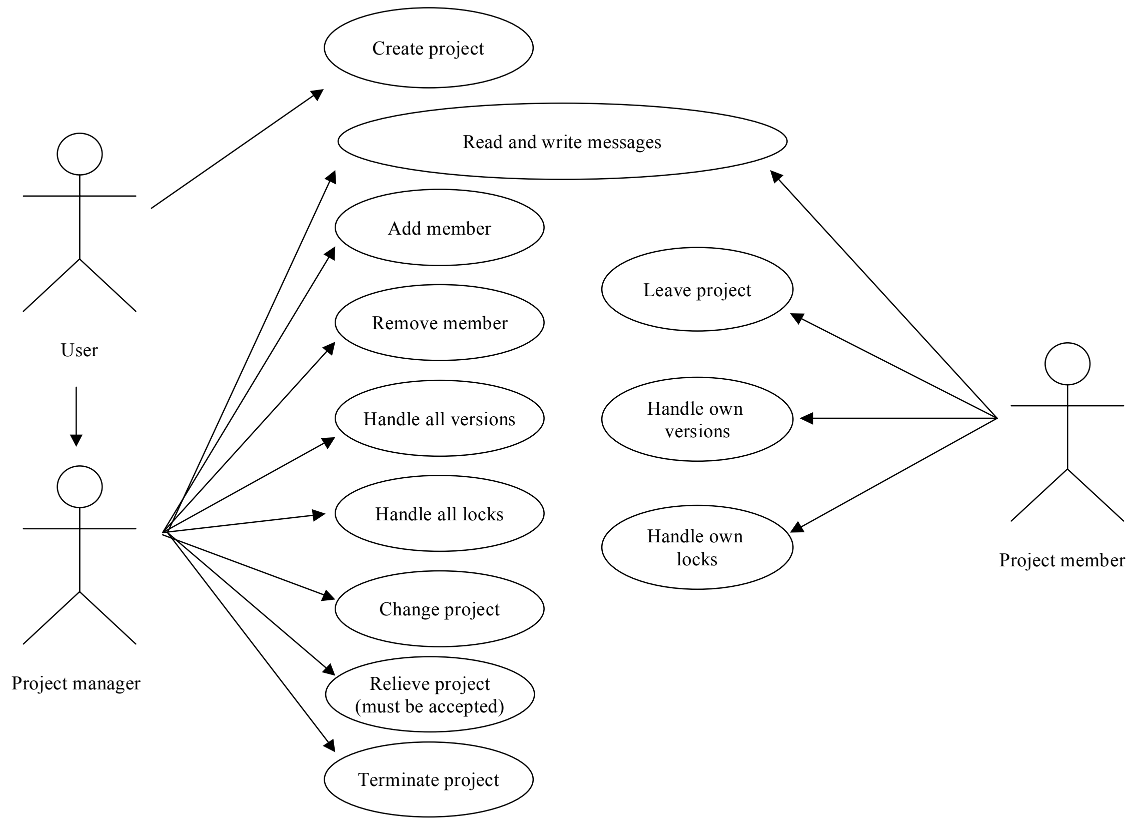


Figure 4.3 Use Cases for the entire project management module

The Use Cases for the entire project module are given a brief description in Table 4.1

Table 4.1 Description of the Use Cases for the entire project management module

Actor	Interaction	Description
User	Create project	Create a project and become the project manager
Project manager	Add member	Add any user as a project member
Project manager	Remove member	Remove any member
Project manager	Change project	Change project properties. This includes the name, description, activating/deactivating file locking, activating/deactivating of version management
Project manager	Handle all versions	Remove any version
Project manager	Handle all locks	Unlock any locked file
Project manager	Relieve project	Choose a member to relieve him of the project manager responsibility. The member then become the new project manager after his approval
Project manager	Terminate project	Terminate the project at any time. All project data, including files, members and project messages will be removed
Project member	Leave project	Leave the project at any time
Project member	Handle own versions	Upload a file into the directory structure, thereby becoming the

		owner of the version of the file. Own versions may be deleted
Project member	Handle own locks	Lock any file that is unlocked, thereby becoming the owner of the lock. Only the owner of a lock and the project manager can remove the lock
Project manager, project member	Read/write messages	Post and read messages in the project forum. Files can be attached to messages

4.4.1 The file manager

The purpose of the file manager in the PM module is to allow the members and the project manager to share and access files within the project in a structured and controlled manner., therefore the following features are supported.

- Uploading of files into a directory structure.
- File locking.
- Version management.
- Strict control so that available project file space is not exceeded.

The Use Cases for the file manager are shown in Figure 4.4

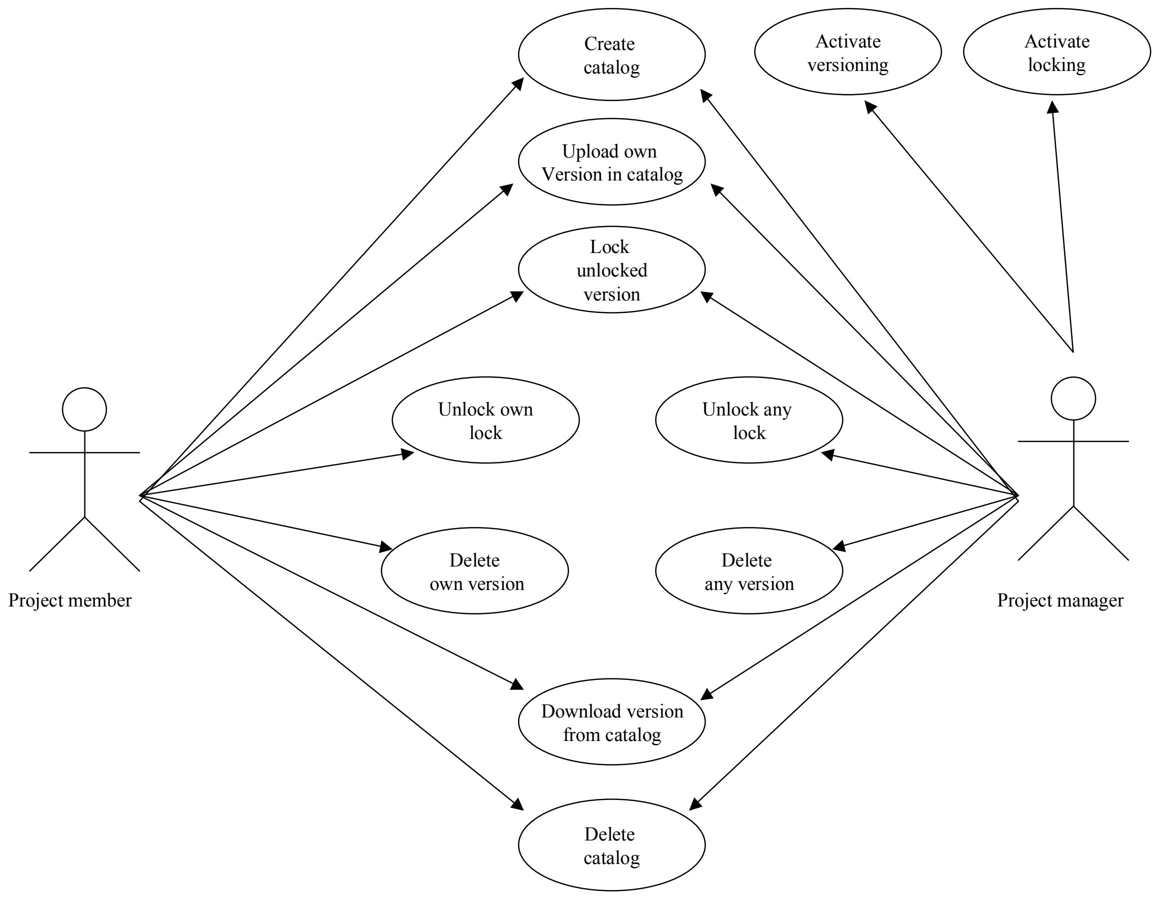


Figure 4.4 Use Cases for the file manager in the project management module

These Use Cases for the file manager are briefly described in Table 4.2

Table 4.2 Description of the Use Cases for the file manager in the project management module

Actor	Interaction	Description
Project manager	Activate locking / deactivate locking	At any time file locking can be activated or deactivated. When locking is deactivated all locked files will be unlocked
Project manager	Activate versioning / deactivate versioning	At any time versioning can be activated or deactivated. When versioning is deactivated all but the newest version of a file will be removed.
Project manager	Unlock any lock	Any locked file can be unlocked, regardless who owns the lock.
Project manager	Delete any version	Delete any version, regardless of who owns the version.
Project manager	Delete directory	Remove the directory and all files and directories in it, regardless of owner and locking status of contained files
Project manager Project member	Create directory	Create a new directory in the current directory
Project manager	Upload own version in directory	Upload a file into the current

Project member		directory. The new version of the file will be owned by the member that uploads it.
Project manager Project member	Download version	Any version can be downloaded, regardless of locking status or owner of version.
Project manager Project member	Delete directory	Remove any empty directory
Project member	Lock unlocked version	Any unlocked version can be locked, thereby the member who locks the version becomes owner of the lock.
Project member	Unlock own lock	Lock any file that is unlocked, thereby becoming the owner of the lock. Only the owner of a lock and the project manager can remove the lock
Project member	Delete own version	Any unlocked version that is owned by the member can be removed.

4.4.1.1 Uploading of files into a directory structure

File transfer to a web server can be done using the File Transfer Protocol, this stores the files directly on the web servers hard disk thereby preventing easy distribution of files over several machines. Therefore we are using the HyperText Transfer Protocol (HTTP) to post the files to servlets, which in turn can store them in a database, which can be distributed, and thereby allowing the files to be distributed.

To perform a file transfer using HTTP from a browser to a web server can be done in three general ways.

- Use a form in HTML to post the file as a multi-part parameter which contain the file name, size and binary stream.
- Let the client download and run an applet. This require that the applet is signed using a public key certificate, and that the client grant the applet access to the client's local file system.
- Let the client download and install a client program, this gives excellent control of the file transfer process. Ofcourse, other protocols can be used than HTTP if using a specialized client software component. This, however, creates problems when crossing platforms if the software is not written in a platform independent programming language.

The solution we have chosen is to do this via an HTML form, with some extensions. The reason that we do not use an applet is that as few applets as possible should be used in order to minimize load times for the client.

It is possible to upload and download files into most relational databases, many implementations support big data blobs of binary data that are treated as a single row, increasing the size of the database file without significant loss in performance.

The files are uploaded into a directory structure, the directory that the file is uploaded to is determined by the path part of the URL of the post request. Each directory has a unique path.

This means that each name within a directory, both filenames and directory names, must be unique. For example, a multi-part post request to the URL

```
http://beta.lecando.com/projectfiles/directory1/directory2?filename=myfile.
txt
```

will store the posted file named “myfile.txt” into directory “directory2” residing in directory “directory1”.

The conceptual representation of directories and files is that of a tree structure, where the directories are nodes, and every file is a leaf as shown in Figure 4.5.

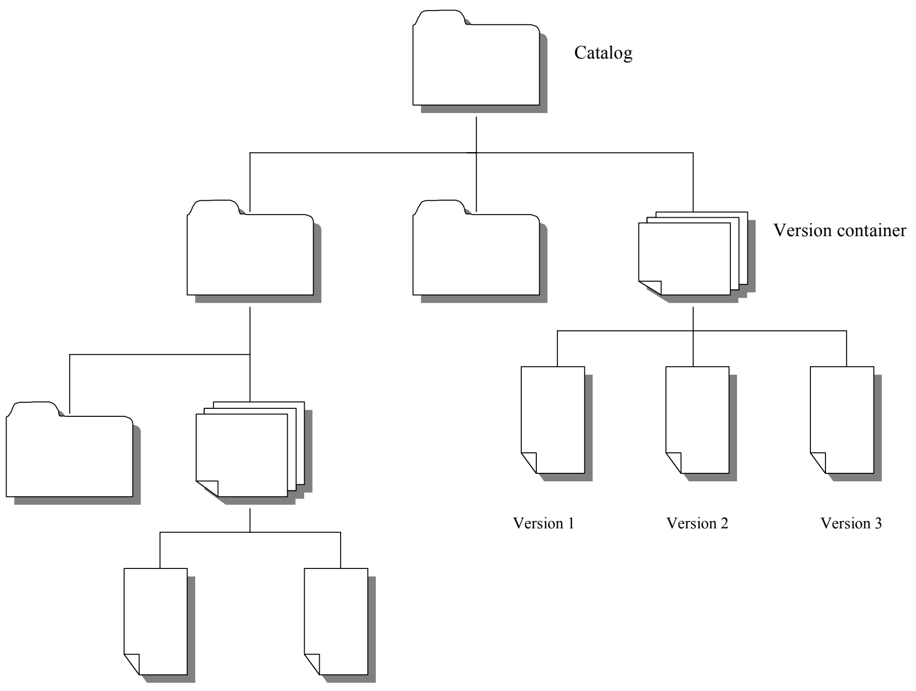


Figure 4.5 Representation of a directory structure in the file manager as a tree.

We believe that the Web based Distributed Authoring and Versioning protocol (WebDAV) will be used for file management in the project manager module at a later stage. WebDAV is further explained in chapter 4.7. To be able to easily support the WebDAV protocol in the future we have implemented most of its features.

4.4.1.2 File locking

The locking mechanism of the file handler is called an exclusive write lock. This means that when a file is locked by a user, no other user can modify or delete the file. Any project member can lock any unlocked file, and the remains until the owner of the lock or the project manager unlocks it. The version handling mechanism creates a new version of the file when an existing file is uploaded, se chapter 4.4.1.3. If a file is locked and the owner uploads a new file, the locking mechanism locks the new file and unlocks the old version. Only the newest version of a file can be locked. The existence of a file during upload is determined as in most file systems, ie pre-existence of the filename in the target path. Old versions can be accessed and deleted, but not locked. When a user uploads a file he becomes the owner of the new version, and only the owner of a version can delete it.

The project manager can set choose to activate the file locking feature when he creates the project. At any time during the existence of the project the project manager can deactivate or activate file locking for the project, if the file locking is deactivated all locked files will be unlocked.

4.4.1.3 Version handling

A file in the file manager of the project management module can conceptually be described as a version container. In the user interface, file operations operate on the newest version of a file by default.

Every file that is uploaded receives a version number. The first time a file is uploaded the version number is set to 1. If the filename of the file already exists in the directory, the uploaded file receives the version number of the existing file plus 1, and the member who uploads the file becomes the owner of this version of the file.

All versions of a file can be accessed by any member of a project, but only the newest version of a file can be locked and unlocked. Any unlocked version can be deleted by the owner of the version, or the project manager.

4.5 The news reader

There are mainly two different types of project systems as discussed earlier. One way is like Microsoft projects, to focus on the project manager. The project system is in one way or the other designed for the project manager to have an overview of what is happening, and how to delegate tasks to achieve its purpose. The other way, the way chosen in this project module, is to let all members be a part of the whole project system. The design is focused on shared information between the members within this project, where all members can take part of that information and work on it. Since the members within a project may or may not be distant, there is an urgent need for a forum of some sort where the members can discuss issues concerning the project. In order to facilitate the communication between the members, a message function has been implemented where the members, as well as the project manager, can read and write messages to the other members of the particular project.

4.5.1 Functionality

The functions required for the news reader were more or less intuitive. Both the project manager and the members within the project have equal rights, and therefor when a member is mentioned in this section, it should be considered to be either a project member, or the project manager. The news reader supports the following features.

- Post a message.
- Read a message.
- Reply to a message.
- List unread messages.

The Use Cases for the news reader are displayed in figure 4.6.

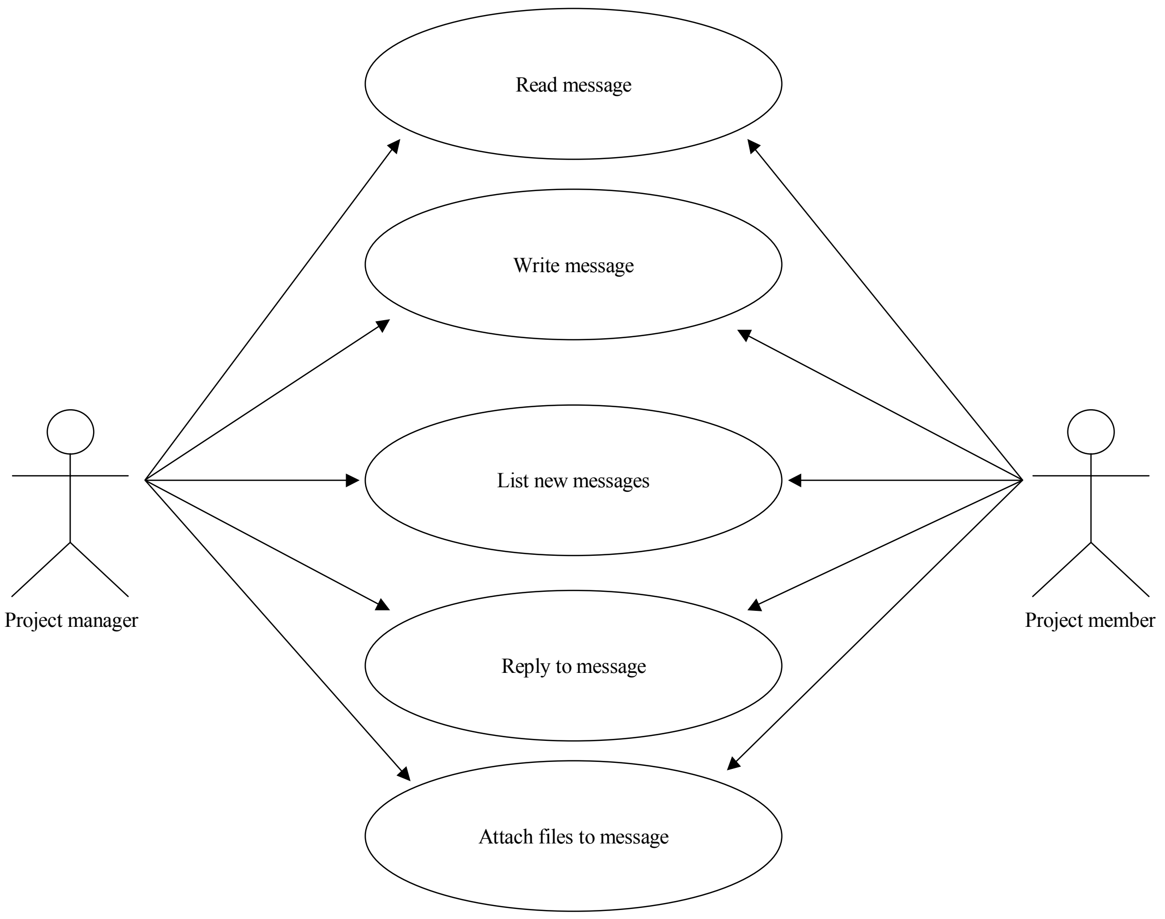


Figure 4.6 Use Cases for the news reader in the project management module

The Use Cases for the news reader are briefly described in Table 4.3

Table 4.3 Description of the Use Cases for the news reader in the project management module

Actor	Interaction	Description
Project manager Project member	Write message / Attach files to message	Post a message and attach any number of files.
Project manager Project member	Read message	Read a message and download message attachments.
Project manager Project member	Reply to message	Reply to any message, including reply messages.
Project manager Project member	List new messages	See a list of messages that the member have not read.

4.6 User interface

The demands on the user interface were quite extensive with web user interface standards. A list of the demands on the user interface is as follows.

- It should be visually appealing and well structured.
- The user interface should be created in a consistent manner with the existing user interface of the other parts of the application.

- All windowed interaction should have a cancel option, including cancelling of bunches of uploaded files and cancelling of added members.
- Notifications of inconsistent file operations should be displayed in a satisfying manner.
- It should both be internationalizable, and localizable to swedish and english
- Buttons should be dynamically altered via context, ie active / inactive
- No logic in JavaScript that is not directly related to presentation.
- No use of applets.
- Different security roles should have different options.

The reason for this limited degree of freedom is that much care have been taken in the other parts of the existing application to have a consistent and logically structured user interface, where certain rules apply to every part of the user interface.

4.7 Web based Distributed Authoring and Versioning

The dominating protocol used in browsers for requests of web-pages is the HTTP protocol, this protocol has seven operations specified, the most commonly used are GET and POST. A GET operation requests a file from a webserver, the file is specified via the URL and possibly some parameters. Typically this file request is of a HTML page and associated files, such as pictures. A POST operation is used to give data to the webserver, typically it is a servlet or a CGI-script that takes care of the posted data and stores it. Forms on a web page often uses POST operations. This protocol is limited and although it supports delete and put operations on files, it does not in itself allow interoperability and collaboration when handling files. There has long been a need to be able to handle files over the internet in a distributed fashion, as can be seen on an intranet where file sharing over the network is transparent to the user .

Web-based Distributed Authoring and Versioning, WebDAV [15], is a protocol that is an extension of the HTTP/1.1 protocol. The stated goal of the WebDAV working group is to "define the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable, while supporting user needs". Be that as it may, the completed features of the WebDAV protocol is:

- Locking: long-duration, exclusive and shared write locks prevent overwriting of files.
- Properties: arbitrary meta-data can be stored for web resources, eg files. Searches can be provided for web resources based on property values.
- Namespace manipulation: web resources may be moved and copied and directories may be created and listed.

As this protocol is based on the HTTP protocol, it receives the benefits of the HTTP infrastructure; strong authentication, encryption, proxy/firewall navigation and world-wide deployment.

This protocol can be used on an intranet as well as over the Internet, as long as the server supports this protocol, and on a WebDAV enabled server, ordinary HTTP-requests as well as WebDAV requests can be handled. The client application must support WebDAV, and as of today there are a number of client products that support this protocol, amongst them are Windows2000, Internet Explorer 5, the Office2000 family, the XML editor Documentor and Xerox Docushare.

The client user can – in a conceptual sense at least – edit documents directly on the server, the downloading to a temporary storage on the client computer is transparent, as is the uploading of the document when saving the file. In Internet Explorer 5 and Windows 2000, it is possible to mount and view web resources such as directories via the ordinary look and feel of the file handling operations of the operating system.

The file manager that we have implemented is developed to be prepared for this protocol, as it uses dynamic path parsing to map requests to file operations and directory content requests.

The file locking operations, as well as version handling can be supported in the file manager of the project management module. The core of the file manager is a servlet that responds to post and get operations that are related to file operations of a project. All files and directories in the project file manager have unique paths, as in an ordinary file system. All that is needed to support the WebDAV protocol is to translate the XML tagged requested operations to the corresponding implemented project file manager operation, and translate the response of the file manager servlet into properly tagged XML.

5 Implementation

In this chapter we discuss how we have chosen to solve the design issues discussed in chapter 4.1 in the implementation of our project management module. We also discuss some practical issues that can affect the efficiency of the development process. In chapter 5.3 we explain responsibilities of packages and key classes in the project management module.

5.1 Design issues

5.1.1 Availability

Regarding improving availability for a server application, such as a J2EE application, a few steps can be taken. Identifying bottlenecks in the system is a good idea, especially those that might make the application crash during high workloads, and to remove these bottlenecks. Regarding identifying bottlenecks see chapter 6.

If it is not possible to eradicate the bottlenecks then availability can be improved by reducing workloads, eg by not allowing more than a fix number of users log on to the system. The maximum number of users is dependent of user behaviour, and can in most cases only be set by a rough assumption of user behaviour. This limit can be altered after monitoring of the server application status have been performed for some time.

5.1.2 Fault tolerance and concurrency

When using a browser as a client, the displayed content of a web page requires user interaction to be updated. This fact makes it very hard to avoid invalid requests to be sent from the client, and therefore forced us to make a large effort in handling errors in a way that it is acceptable for the user. The file handling components of the project module will generate numerous invalid requests if several users attempts to lock, unlock or delete the same file. Even though the underlying data handling can handle concurrent modification of data in a consistent manner, for us it was necessary to detect these inconsistencies before attempting to update the data structures. Locking an already locked file might seem like moot point, but since the lock is owned by the user who has locked it, this will be advert behaviour. Instead errors of this type are handled by displaying available information of the error in a popup window to the client that sent the invalid request.

For example, user A might have received a web page displaying file system status of a project at time X, and user B locks the file F in the same project at time X+1. User A sends a request to lock file F at time X+2. Then the request of user A will be that of locking an already locked file, this inconsistency is detected and A receives a message “File F is locked by B since time X+1”. There are a number of cases where these inconsistencies can occur, and we have tried hard to identify all of these cases and reduce their impact on the users.

5.1.3 Scalability and performance

One of the main objectives for the development of this module was to create a module with high performance, and being scalable as well. A J2EE server application shall not be restricted on running on one server, and ideally it should be able to distribute the EJB components over several servers. To use several servers for one application is called clustering of servers. At the time of the writing of this report, clustering has not been done with the J2EE application so nothing conclusive about the possible difficulties and effects can be said about this. However, the J2EE server used, Orion server, does not support distribution of the underlying database on several servers at the time of writing of this report.

In order to optimize performance, EJB references should be kept to a minimum, since the lookup of EJB bean takes an average of 20 ms on a high-performance server. To minimize EJB lookups required for a request, it is better to keep as much data as possible in every entity bean type. Thereby instead of having a multitude of different entity beans which forces the J2EE server to do several lookups in order to retrieve the wanted data, one EJB lookup might be sufficient.

It is also possible to minimize lookups by using a mirroring java class for every entity bean, a Value Object [21] that have the same attributes as the corresponding bean, and methods to support retrieval of the attribute values. When a method shall retrieve data from a bean, it does not return the bean object, instead the method instantiates a value object with the data from the bean, and returns the value object. The value objects are mainly used to hold data to be accessed in the presentation layer and shall never be stored anywhere, since an altering of an entity bean does not alter the already instantiated value objects for this bean.

These solutions in order to improve performance conflicts somewhat with the paradigms of object orientation, but we firmly believe that it is better to be pragmatic under certain circumstances.

5.1.4 Security

Authentication is the procedure of verifying a claimed identity. In the J2EE application that our module is a part of, authentication is handled via HTTP Basic Authentication. This requires the browser to send an autorization header (which rather should be called an authentication header) with every request, containing user login and password. When the browser connects to an URL that requires authentication for a realm, it handles this via a popup window asking the user for login and password, and then this information is kept by the browser for every request within the realm. The authorization header is encoded with base64 encoding, a simple algorithm that should not be regarded as secure.

There are two other ways that the authentication can be handled in a J2EE server, HTTPS Client Authentication and Form Based Authentication.

When the authentication is successfully done, the J2EE container establishes a login session with the user, sessions are discussed in 5.1.5.

The authorization model that is used in our module is role-based, where the different roles and their rights on project level are:

- Non-member – no rights to do anything within the project.
- Member – use the file manager, post and read messages, leave the project.
- Project manager – use the file manager, post and read messages, add / remove members, terminate a project, hand over the project management to a member.

- Head administrator – same as the project manager except handing over a project.

This role-based authorization is implemented on method level in the service layer, meaning that authorization checks are done in the service layer before any action is taken.

5.1.5 Session

A J2EE server creates a client session a client has provided a valid authentication with a request, and the session is lost after a period of time if no new requests are made. This time period is around half an hour in the Orion server version. The client session can be kept in three different ways.

- HTTP cookies, which basically is a lookup-table in the browser containing server specific information. A J2EE session cookie contains a reference number that the J2EE container can retrieve from the browser, and this reference can be mapped to client data on the server.
- URL rewriting, which can be used when the client will not accept a cookie. The J2EE serve adds client session reference number to URL requests within the domain as a path parameter.
- SSL session, the HTTPS protocol supports identification of a client as being part of an accepted session.

The application that contains this module uses cookies to identify client sessions. In the session, attributes can be stored and retrieved. Attributes can contain any Java object, and these are retrieved and set with a name as a key. In both servlets and JSP pages the session attributes can be retrieved, altered and set.

5.2 Development issues

5.2.1 Visual development tools vs. texteditors

To work with the code of a large java application can become rather cumbersome. There are a number of packages and a large number of classes. So, when writing new code, reusing existing code or modifying code, it is not feasible to remember where all methods and classes are declared and used, and as a developer it is a of great help to file searching and scanning tools, such as Linux / Unix “find” and “rgrep” shell commands.

When we started out on this project, we were contemplating to use a visual development tools, such as Visual Age or Forte. Both are powerful with a lot of functionality to aid the developer, such as graphic tools for building of user interface, lookup of where methods are declared and used and versioning management. However, we chose not to use any other tool than the built-in tools of Linux, eg the Emacs text editor and CVS. The reason for this was multifaceted, but mainly due to the insufficient performance of our workstations, not being able to run these environments at fully acceptable speed.

5.2.2 JUnit

We have implemented unit testing of our code with the aid of JUnit, and the time it takes to write tests is well spent. One possible drawback of using JUnit is that it becomes slower to make changes in the existing API, since this requires changes in the testcode as well. This might be a good reminder of keeping the API backwards compatible, however. While using JUnit the thought of having a testtool for the user interface comes to mind. Since JUnit detects many errors at compiletime, a substantial part of the errors encountered are in the user interface, not in the business logic. However, automated testing of the user interface is complex and requires much more work initially than component testing.

5.2.3 Concurrency Versioning System, CVS

We used the Concurrency Versioning System shipped with the Red Hat Linux distribution. This is a powerful and versatile tool that mainly helps with two tasks, structured multi-user modification and merging of files and an incremental backup system.

It is set up by specifying a repository directory on the local file system or on a remote files system. This repository is the centralized storage area for files. When a file is edited, it can be copied to the repository, where other users can access them. CVS automates much of the copying tasks, and also has strong integrity checks to prevent accidental misuse. So CVS is, apart from the user interface, an extremely versatile tool that makes it possible to work with shared files in consistent manner.

5.2.4 Javac compiler vs. Jikes

When we made changes in our code, even the JSP pages, we had to recompile the entire server application to be certain that the changes was reflected in the deployment of the application. This compilation could take anything up to seven minutes, and this was after we had switched to the much faster java compiler Jikes, which outperforms the standard Javac compiler from SUN with a time factor of about 10.

5.2.5 Networking File System, NFS

We kept our application files in our NFS mounted home directories, and this was modus operandi for the development team. This had some advantages, such as having the ability to sit down and work on any computer in the network. Unfortunately no work could be done when the file server crashed, and this happened on numerous occasions due to poor dimensioning of the file server hardware. The CVS repository was kept on a server residing outside the internal network, and disruption of access to this server also impaired the development work.

5.2.6 Synergistic effects of teamworking

There are several advantages of working together in a closely knitted team of two persons as we did during this project. Learning and understanding documentation, brainstorming, error corrections, discipline to follow procedures, systematic work – simply put the quality is improved due to the fact that two brains outsmart one. However in terms of development speed, it is rarely the case that two persons develop twice as fast as one. Quite often when we both had to modify or add methods to the same class, the work was halted for one of us. Nonetheless, low quality generates much extra workload in the long run, and this has to be accounted for while summing up the work.

5.3 Classes and packages

5.3.1 Overview of classes

It is possible to categorize classes for our module into three categories:

- General project components.
- Specific file handling components.
- Specific project message components.

For detailed information regarding classes and packages, please see the appendix for Javadoc-generated documentation.

5.3.1.1 Servlets

The J2EE specification does not demand support for HTTP post operations to JSP pages, but Orion server supports this. In order to be J2EE server independent, no HTTP post operations to JSP pages are done, instead all posting of data are sent to servlets.

5.3.1.2 General project components

Represented mainly by the package *com.lektor.project*. Here the entity bean ProjectBean and it's related interfaces are defined, a ProjectBean contains all data regarding the project frame, such as name, description, project manager and member list.

The session bean ProjectServiceBean and it's related interfaces are also defined here, and the responsibility of this component is to do authorization checks of the client's before a method is invoked on the Project remote interface.

There is a number of servlets in *com.lektor.project.servlets* used to process posted data from client browsers, and then invoke methods in the ProjectService remote interface. These servlets does not invoke the ProjectService methods if the posted data is apparently invalid, but instead passes on error information back to the client, where it is presented.

5.3.1.3 Specific file handling components

Represented mainly by the package *com.lektor.project.projectfile.* Since J2EE does not explicitly support filestreams, we use SQL statements to put the binary filestream into the database. Unfortunately the SQL language is not entirely standardized, thus making some SQL statements database specific. However, a list of database queries, as well as runtime lookups of what database is being used provide the means to use a new implementation of a database with only minimal modifications of the code.

The class ProjectFile represents a system unique file pointer, and also contains file status such as locking status, owner and creation date. A couple of helper classes retrieves or stores the binary filestreams. The session bean ProjectFileServiceBean and related interfaces are also defined here, and it coordinates file operation tasks after authorization and consistency checks have been performed.

The servlet ProjectFileServlet is used for HTTP get requests for folder information and retrieval of files, as well as HTTP post operations to perform file system modifications, such as retrieving or deleting a file. This servlet listens to all requests beginning with a specific path, and here the rest of the request path determines the corresponding file path. For example a get request to *http://.../projectfiles/project1/* will make the servlet return a collection of all folders and files in the root folder of the project with id 1. A get request to *http://.../projectfiles/project1/ folder1/folder2/file1.txt* will give a servlet response containing file file1.txt in the directory /folder1/folder2/ in the project with id 1. All operations on files uses dynamic URLs to give the ProjectFileServlet information of the corresponding file paths.

The servlet UploadFileServlet supports storage of file objects in the session, thus making it possible for the user to upload several files into the session and, thus giving the user an option of cancelling the operation. If the users posts a save request, all files in the session are stored in the project file system.

5.3.1.4 Specific project message components

Here we added classes and servlet classes to existing packages, *com.lektor.message* and *com.lektor.message.servlets*. We implemented in compliance with an existing message API and an existing user interface. And modified a few existing methods to include the project messages. The existing session bean MessageServiceBean does the authorization checks and invokes methods on appropriate EJB components. The ProjectMessageBean supports threading of messages, each reply message stores information about the message it is a reply to. At the time of the writing of this report the user interface does not support this.

To support file attachments, a couple of servlets have been implemented that also uploads files to the session to give the user a chance to cancel the operation.

6 Load testing

The purpose of load testing a distributed system is to gain insight of its behaviour during various workloads. Load testing can be used during development to bring design flaws up to surface. It is also a tool for asserting that the system meet specified requirements.

In the Internet industry, availability is often of great importance, and this is what we aim to test, workload performance, ie how the response time for the server varies with workload. We want to measure response time under different loads.

In order to perform relevant load tests one must have some knowledge of how the system will be used. In some cases this can be well defined, a system that is developed for one customer and is specified from its intended use is an example of this. However more often it is hard to predict the use of a system. A system that is exposed to a number of different of users will almost certainly be used by some users in a way that the designers did not have in mind.

For our system we have little information about the users but the functionality of the system give us some insight to perform relevant tests.

6.1 Tools

In recent years there has evolved a variety of test tools for workload performance, such as Siteload. They can be set up with a number of test requests for a web server, and then it is possible to simulate a number of users making the requests while measuring response times. Since these tools are expensive, Siteload cost around 10.000 EURO for a license to simulate 100 concurrent users, it has not been possible for us to use such a tool.

In order to perform our load tests we have developed a small test tool. The test tool is written in Java and starts a thread for every user it shall simulate. Each thread receive a valid and unique login and password to be used during the requests. Each thread also receive information of which project the requests shall concern. The threads prepares a specified request dynamically with regard to login, password and project. When all required threads have been created, the threads open the HTTP connection they need in order to perform the request. Then they measure the time elapsed from the start of the request until the entire response have been received. The request can be repeated a number of times or done once, every time measure is reported to a separate synchronized thread that is responsible for storing results from all threads in a file.

In reality a server handle requests from several client machines as shown in Figure 6.1.

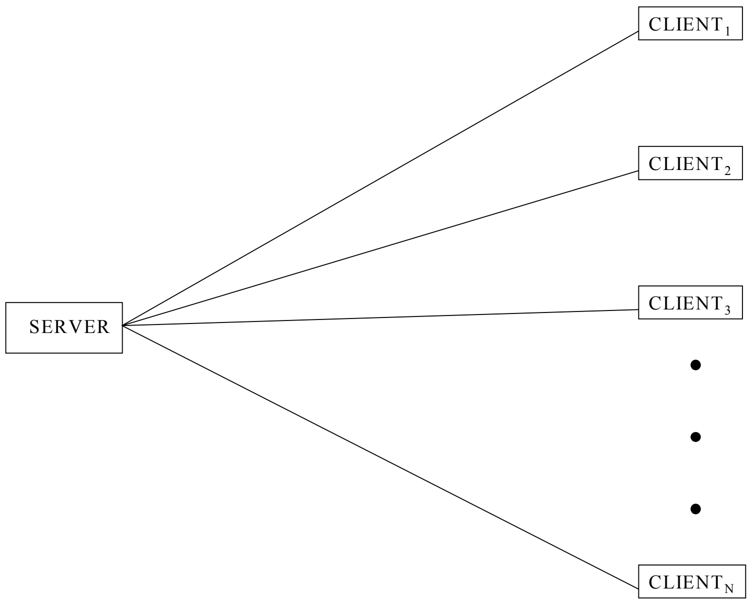


Figure 6.1 Several clients accessing one server simultaneously

We use one machine to simulate several clients by the use of threads and one thread represents one client as shown in Figure 6.2.

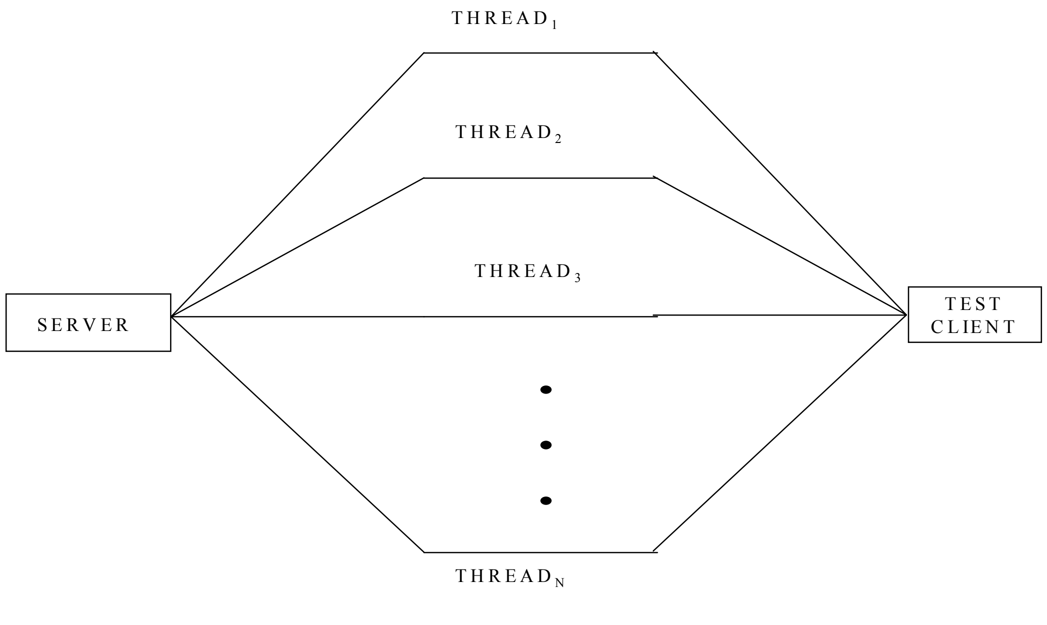


Figure 6.2 Simulation of simultaneous access from several clients using one client machine

We did not perform any tests on how accurate this representation is.

6.2 The test bed

The configurations shown in Table 6.1 were used during the tests.

Table 6.1 Test bed

Server	Testclie
Pentium II 400 128 100 SUN JDK Orion Solid Embedded	Pentium II 350 128 100 SUN JDK n/a n/a

6.3 Test strategy

To perform tests in a larger scale one need appropriate tools to set up the test environment. A tool for automatic creation of a test database is required to measure performance with large databases. Since we did not have access to any such tools we had to confine our tests accordingly.

We did not have any means to create a test database other than by manual labour, either by typing SQL queries or by using the server application itself to create testdata from an empty database. We therefore decided to limit our tests to requests that trigger read operations on the database. Furthermore we decided to perform our tests on only one small database. The database used had the size shown in Table 6.2.

Table 6.2 Row size in database used during test

Table / Entity Bean	Rowsize
project	5
projectMessage	25
projectMembers	200

On ad hoc basis we decided that we would not perform any tests with more than 30 simulated clients. This was due to the fact that we performed all of our requests from one machine and we have no real insight in how representative this is.

In order to perform tests on different aspects of the system we chose the following four requests:

- *Properties*, lists all project members in one project.
- *New messages*, lists all unread messages for all message forums, for a user.
- *Message list*, lists all member messages in one project.
- *Project list*, lists all projects that are accessible for a user.

We used response time, the time elapsed from that a request is sent from a client until the client receives the server response, as one measure during our tests. Furthermore we measured the total time elapsed until all requests had been responded.

With decisions on the database size and request types made we set up the following test schedule.

- Perform each request in an undisturbed environment for use as reference value.
- Perform each request from several simulated clients and observe the impact on response time and total time.
- Perform all requests from several different clients and observe the impact on response time and total time.

The result from our test follows.

6.4 Test results

6.4.1 Reference values

To obtain reference values for each request we initiated ten separate requests for each request type, then we calculated the mean and standard deviation for the response time for each request type. The results are shown in Figure 6.3

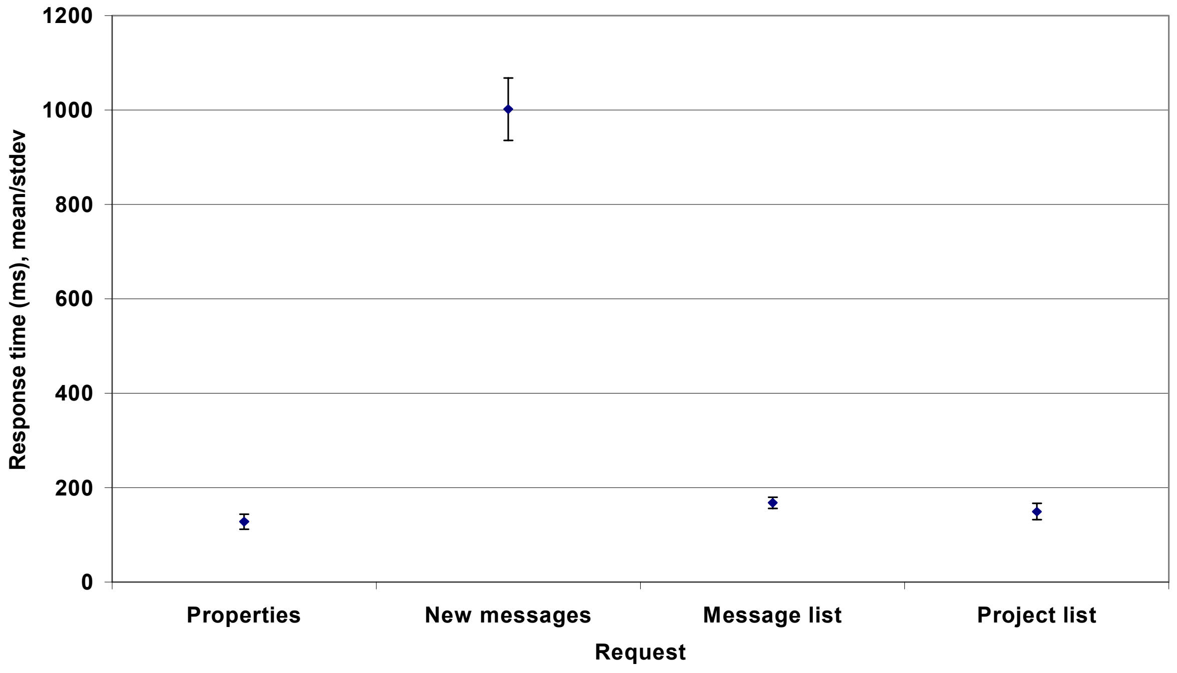


Figure 6.3 Response time by request - reference values, request performed with no other load

We can observe that the standard deviation for the response time for each request type is small in magnitude, this implies that the mean values for the response time for each request type will serve well as reference values.

6.4.2 Separate requests

These tests were conducted as follows; for one request type we simulated a number of clients that each performed the same request type. The request was performed with different parameters to avoid caching in the server, each request required a separate lookup in the database.

Our intention was to simulate 5, 10, 20 and 30 clients for each request type and observe the effect in mean response time and total response time. Unfortunately the server application crashed on several occasions during our test, this happened for request types that we know put the application at great strain with increased the workload. Therefore the *New message* request could not run for more simulated clients than 10 and the *Project list* request could not run for more than 20 clients.

We could not establish the reason behind the fact that two of the queries did not run for 30 simulated clients. The problem could either lie in our client software or on the server side. If the problem resides on the server this is a serious issue that needs attending to.

With more time at hand and proper tools we would have investigated this further.

The result for each request type follows.

6.4.2.1 Request: Properties

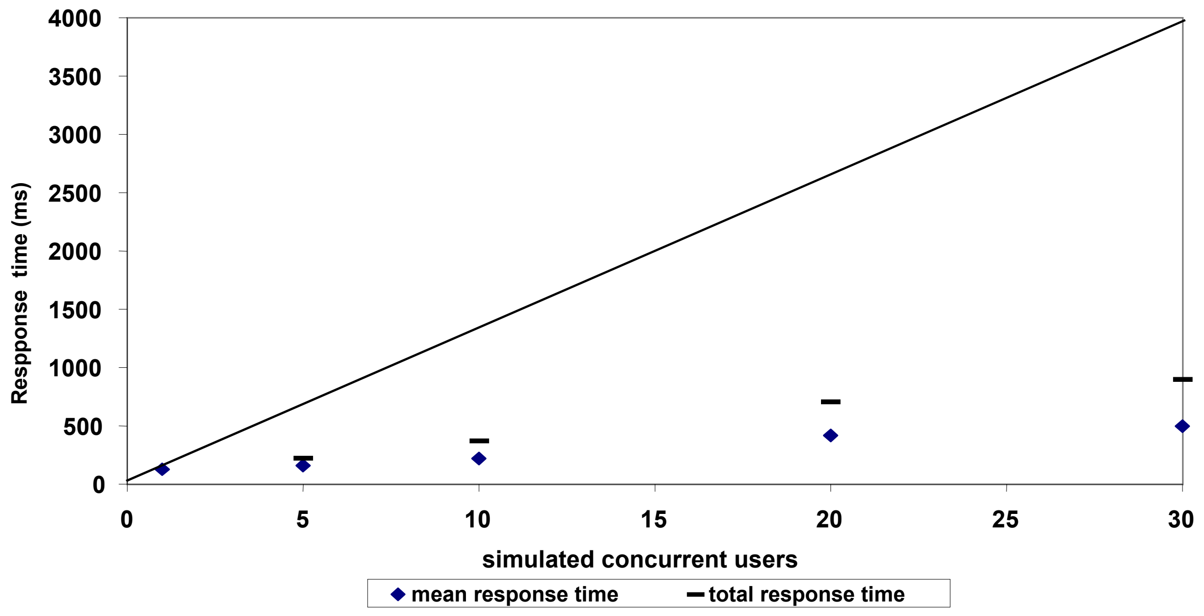


Figure 6.4 Simulated request: Properties

In the graph above the results for request *Properties* are shown. The horizontal marks represent total time for all requests. For reference the line representing sequential handle of responses is added. It should be noted that the response time consists of time for activities outside the server (network), a total response time close to the reference line indicates poor behaviour from the server.

The mean response time is 500 ms for 30 concurrent requests, this value in itself could not be considered bad.

The total time for each simulation is well below the reference line.

6.4.2.2 Request: New messages

We could only simulate 10 concurrent users for the *New Messages* request. We note that the mean response time increases dramatically with increasing load. The total time for the requests is on or close to the reference line. This indicates that the server handle all requests in sequence. The fact that the mean response time as well is close to the reference line is alarming. The mean response time for ten requests is 7 seconds. If the server can respond to one request in 1 second (the reference value) the strategy of sending the requests in sequence would lead to a mean response time of 5,5 seconds. This behaviour of the server application is at least poor. The results for this test is shown in Figure 6.5.

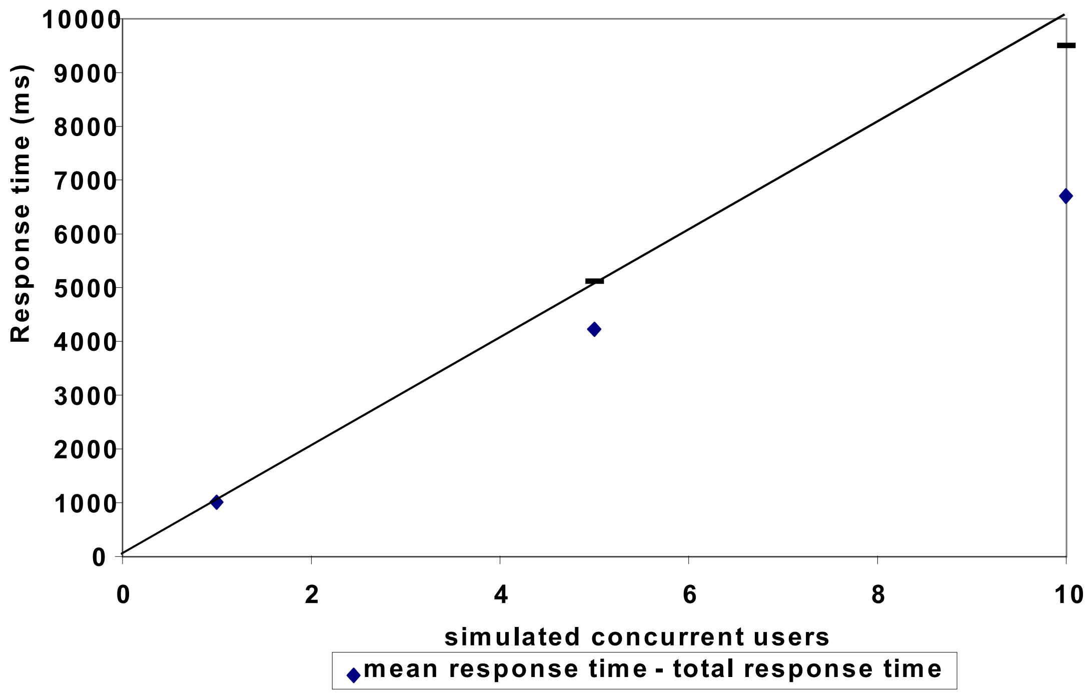


Figure 6.5 Simulated request: New messages

6.4.2.3 Request: Message list

The result for *Message List* is similar to those from *Properties*. The mean response time for 30 requests is well below one second, the results are shown in Figure 6.6.

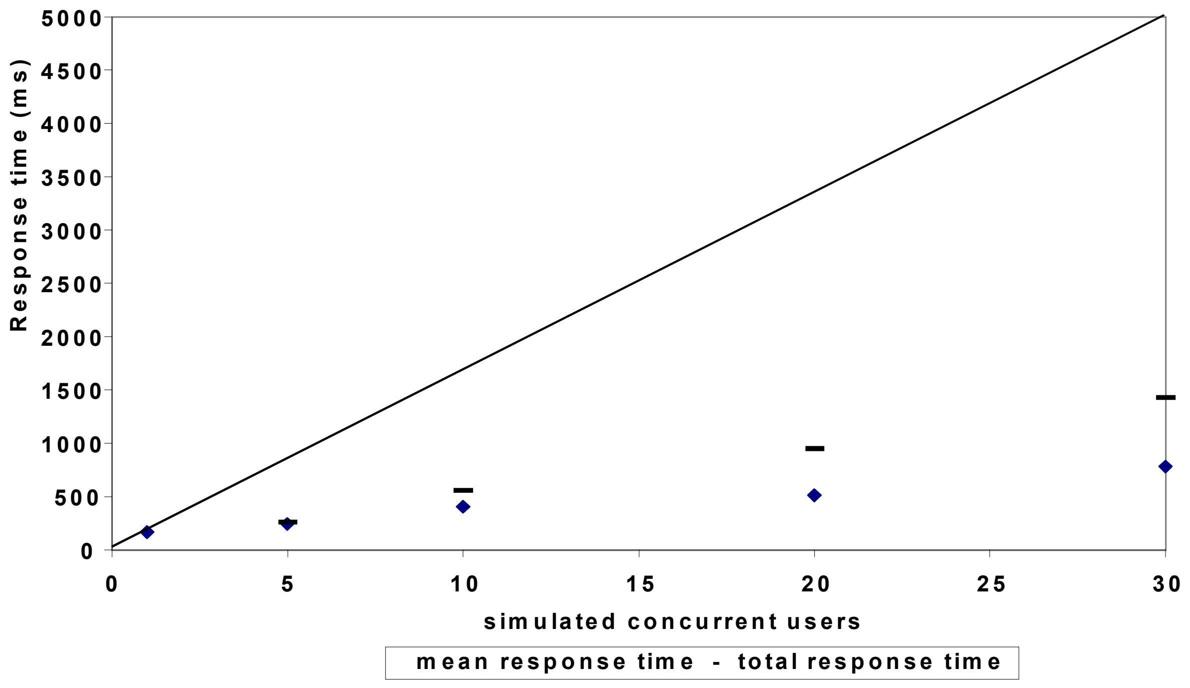


Figure 6.6 Simulated request: Message list

6.4.2.4 Request: Project list

We could only simulate 20 concurrent users for the *Project List* request. Otherwise the result is similar to those of *Propertie*s and *Message List* with a little steeper increase of mean response time with increasing load. The results are shown in Figure 6.7.

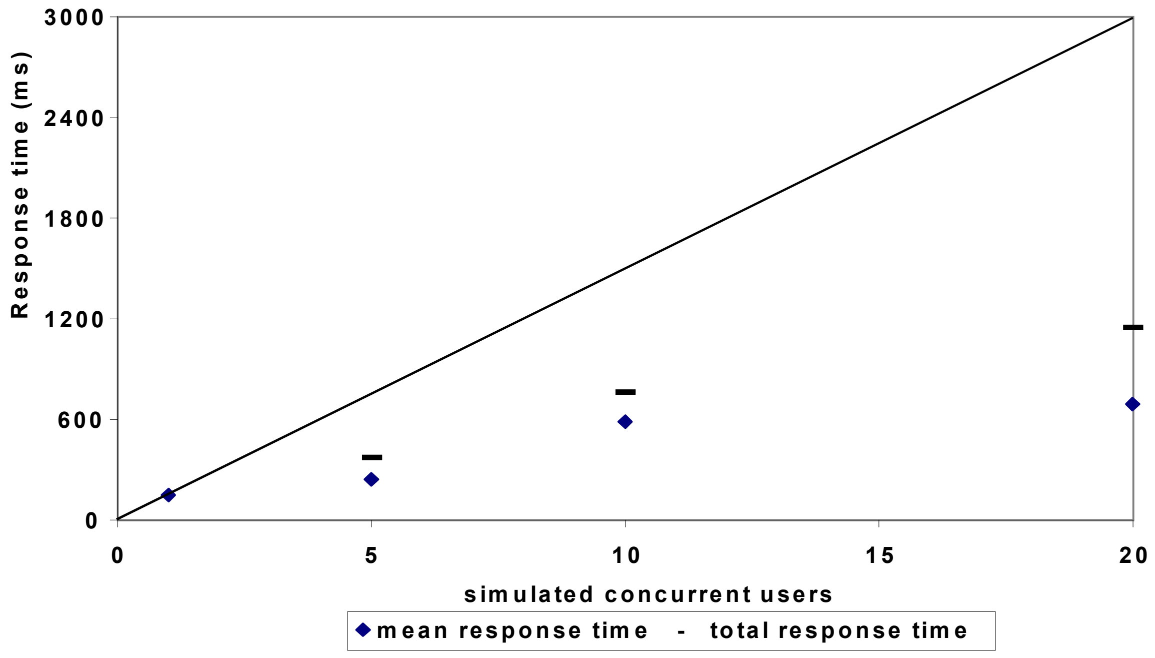


Figure 6.7 Simulated request: Project list

6.4.2.5 Mixed requests

In this test we made two simulations. One with 15 simulated clients and one with 30, as shown in table 6.3.. In each simulation we let clients perform one of the following requests in a uniformly distributed manner; *Properties*, *Message list* and *Project list*. We deliberately chose not to include the request *New messages* considering the problems that occurred in the previous simulations.

Table 6.3 Distribution of request type in simulation of mixed requests

Simulated clients	Properties	Message list	Project list
15	5	5	5
30	10	10	10

The results from the simulations follow.

The mean response time increases with increasing load as shown in Figure 6.8. This is not remarkable in itself but an increase in mean time for the *Message List* request of approximately six times is somewhat alarming. We crosschecked the results from these two simulations with the previous simulations with separate request types.

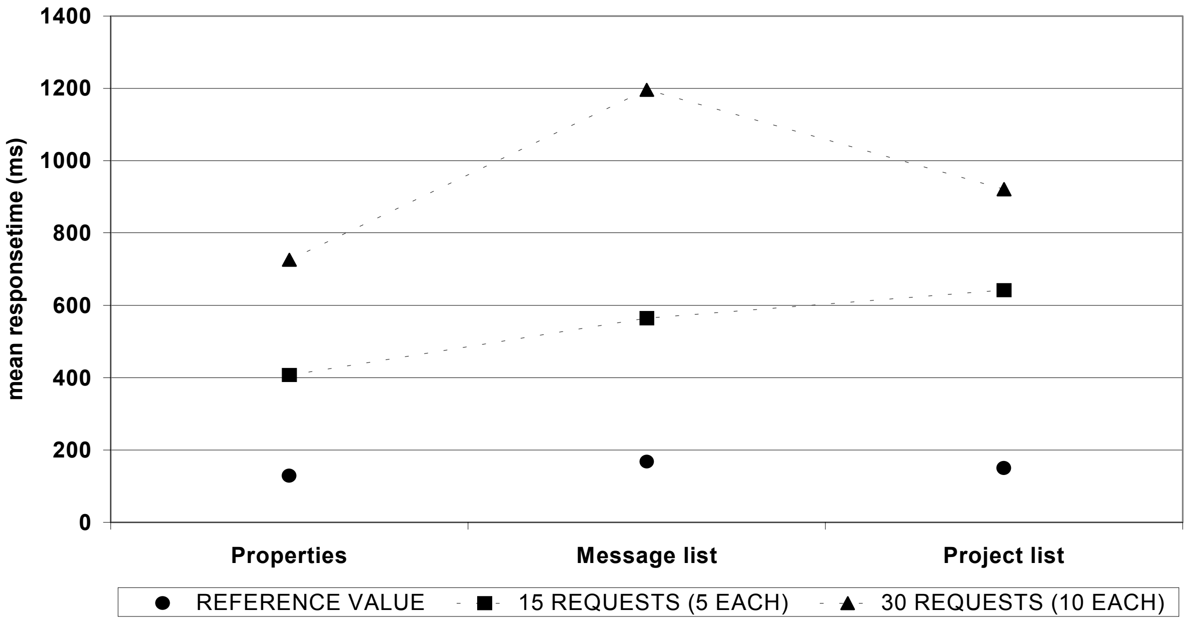


Figure 6.8 Mean response time in mixed request simulation

In figure 6.9 we compare the total response time for several simulations. In comparison the mixed request simulation of 15 clients needs more time than any of the requests needs to process 20 of its own. The mixed 30 clients test lacks comparison with *Project list* but is well above the other two on the same number of simulated clients.

This implies that the server has some cost in time when handling different requests

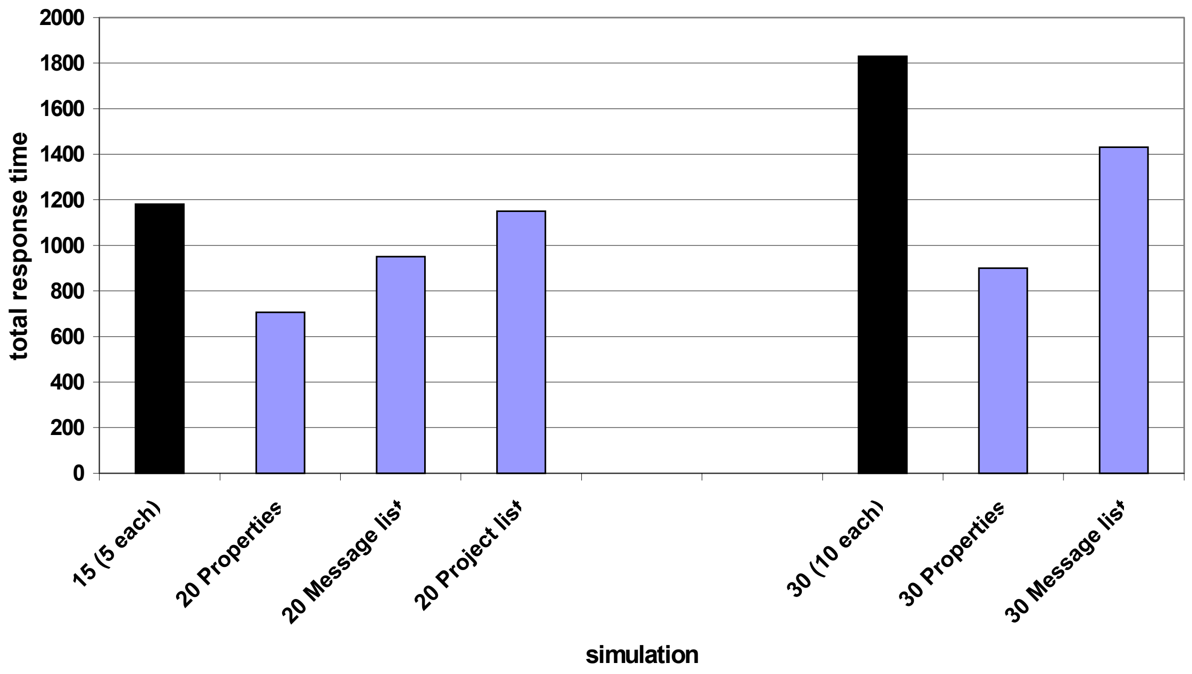


Figure 6.9 Comparison between mixed and separate request simulation

6.5 Conclusions

For similar levels of workloads it was possible to find differences in mean and total response time while comparing single request type to mixed request types. This difference suggests that the web server uses a cache mechanism of data that is more effective for a single request type than for several. This is most likely due to the fact requests of the same type gather data from the same type EJB components, and these can be cached by the web server.

Our tests clearly indicate that the request to list new messages in the news reader perform inadequately, we could not even run our small test suite for this request type without suffering from a server crash.

It should be kept in mind that the web server used during these tests were a substandard desktop workstation, since we did not have the access to any high-end webserver to perform the tests on. All the threads to simulate users were run on another substandard desktop workstation. Regarding performance differences between the desktop workstations we used and standard web server hardware it is adequate to say that there is a performance gain using a high-end webserver. Since this difference is difficult to quantify we will not delve into more detailed comparisons than that.

To perform load tests is a complex, multifaceted task and it is not within the scope of this master’s project to perform more complete load tests than we have done. Several questions remain unanswered.

- The database size was small and this affected the lookup time in the database for the requests. With a larger database, could we have provoked different behaviour in the system?

- The test suite was small, partly due to substandard hardware. How would larger sets of requests and simulated users have affected response times?

As mentioned, performing relevant load tests is not an easy task. It requires:

- Appropriate tools for the setting up of test environment.
- Representative means for simulating several clients from one machine.
- A thorough test strategy and time to conduct it.
- Methods for appropriate analysis of the results.

7 Conclusions and future work

7.1 Summary

We have developed a project management module using distributed technology platforms. We have identified parts of the module with possible low performance by conducting and analyzing load tests on the module.

In order to keep the time plan, we adopted an incremental approach for our development. Initially we implemented the base functionality, and then assessed what other functionality could be added within the scope of the remaining time. Unfortunately we overlooked the complexity of implementing an acceptable user interface for our module, as well as the need to rewrite non-trivial parts of the underlying code to meet the demands for the user interface. This made the time plan hard to keep, but not impossible since it was kept.

It is our firm belief that it is more efficient to first implement a detailed user interface, or at least decide on a detailed level how the system shall interact with the user and vice versa. The benefit of this approach is that you then have a detailed specification of exactly what functionality have to be supported and implemented, making it easier to make approximations for time required.

7.2 Suggestions for future work

7.2.1 Improvements to the file manager

There are a few features that could be implemented in the file manager that would be useful for the end user:

- Copy function of files.
- Copy function of folders.
- WebDAV support.

7.2.2 Improvements to the news reader

The news reader is a possible bottleneck, se chapter 6. It is therefore suggested to optimize this on a system level, not only the project message part of the news reader.

7.2.3 Other functionality

- Calendar function, where it is possible to keep track of and schedule meetings. The complexity of this function depends largely on the user interface, it is not complex to implement support for this in the core of the module.
- Notepad function, an easy way to create documentation and similar documents in a structured manner. Something similar to a Wiki [17].

- Message system, separate from the news function, making it possible for members to post project related messages not suited for the news reader of the project.

8 Appendix

8.1 References

[1] Goodman, Danny. 1998. *Dynamic HTML The Definitive Reference.* ISBN 1-56592-494-0, Sebastopol: **O'Reilly** & associates inc.

[2] Flanagan, David. 1998. *Javascript The Definitive Guide.* ISBN 1-56592-392-8. Sebastopol: O'Reilly & associates inc.

[3] Fowler, Martin & Scott, Kendal. 1997. *UML distilled applying the standard object modeling language.* ISBN 0-201-32563-2. Massachusets: Addison-Wesley Longman inc.

[4] Gollmann, Dieter. 1999. *Computer security.* ISBN 0-471-97844-2. New York: John Wiley & sons.

[5] Kassem, Nicholas. 2000. *Designing Enterprise Applications with the Java™ 2 Platform, Enterprise Edition.* ISBN 0-201-70277-0. Palo Alto: Sun Microsystems inc.

[6] Monson-Haefel, Richard. 1999. *Enterprise JavaBeans™.* ISBN 1-56592-605-6: Sebastopol: O'Reilly & associates inc.

[7] Monson-Haefel, Richard. 1999. *Enterprise Java Beans specification v1.1*

[8] Eckel, Bruce. 2000. *Thinking in Java.* ISBN 0-13-027363-5: New Jersey: Prentice-Hall

[9] Shannon, Bill. 1999. *Java™ 2 Platform, Enterprise Edition Specification, v1.2.* Palo Alto:SUN Microsystems inc.

[10] Coward, Danny & Davidson, J Duncan. 1999. *Java™ Servlet Specification, v2.2.* Palo Alto: SUN Microsystems

[11] Cable, Larry & Pelegri-Llopart, Eduardo. 1999. *Java Server Pages™ Specification.* Palo Alto: SUN Microsystems

[12] Object Mentor inc. *Junit Cookbook,* <http://www.junit.org/junit/doc/cookbook/cookbook.htm> 01 Oct. 2000

[13] Object Mentor inc. *Infected: Programmers Love Writing Test,* <http://www.junit.org/junit/doc/testinfected/testing.html> 03 Oct. 2000

[14] Swedish Institute of Computer Science. *The Mozart Programming System,*<http://www.mozart-oz.org>

[15] Stein, Greg. *Welcome to WebDAV Resources, < http://www.webdav.org>,* 06 Oct. 2000

[16] IETF WEBDAV Working group. *World Wide Web Distributed Authoring and Versioning.* < http://www.ics.uci.edu/pub/ietf/webdav/> 15. Nov 2000

[17] WikiWikiWeb. *Welcome visitors,* <http://c2.com/cgi/wiki?WelcomeVisitors>

[18] Microsoft Corporation. *Microsoft Office*, <http://www.microsoft.com/office/project>

[19] Microsoft Corporation. *Microsoft Office,* <http://www.microsoft.com/office/outlook>

[20] Projectplace. *Projectplace.com,* <http://www.projectplace.com>

[21] Ubilab, *Values in Object Systems,* <http://www.jvalue.org/papers/ubilab-tr-1998-10-1.pdf>

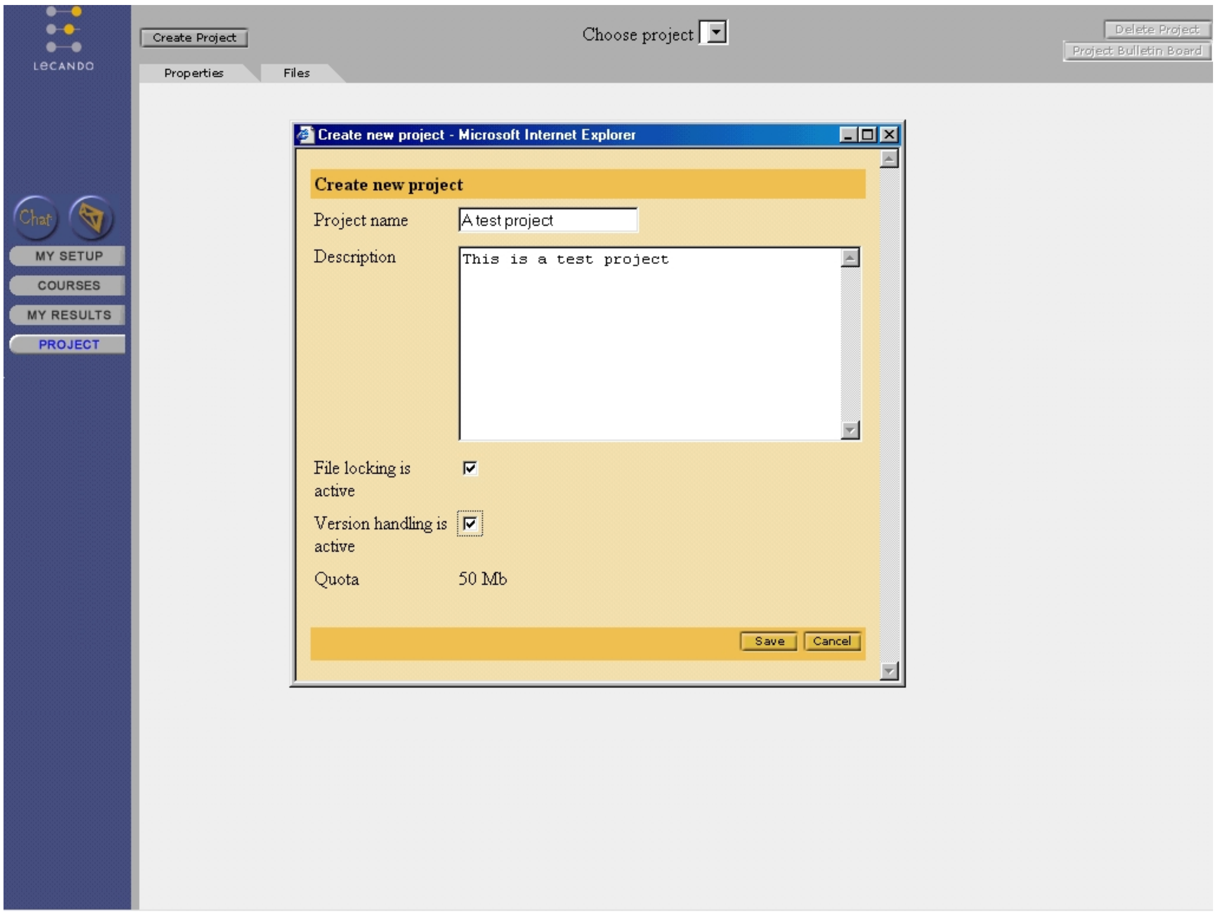
[22] Vlasov, Vladimir. *Distributed Computing in Java. Java and CORBA. Java RMI,* <http://www.it.kth.se/edu/gru/Java/2000_2001/lectures/index.html>, 01 Dec. 2000

[23] Vlasov, Vladimir. *Overview of the Enterprise Java Technologies: JDBC, Servlets, JSP, EJB, JNDI ,* <http://www.it.kth.se/edu/gru/Java/2000_2001/lectures/index.html>, 01 Dec. 2000

8.2 Examples from the user interface

Here we illustrate the following user interactions.

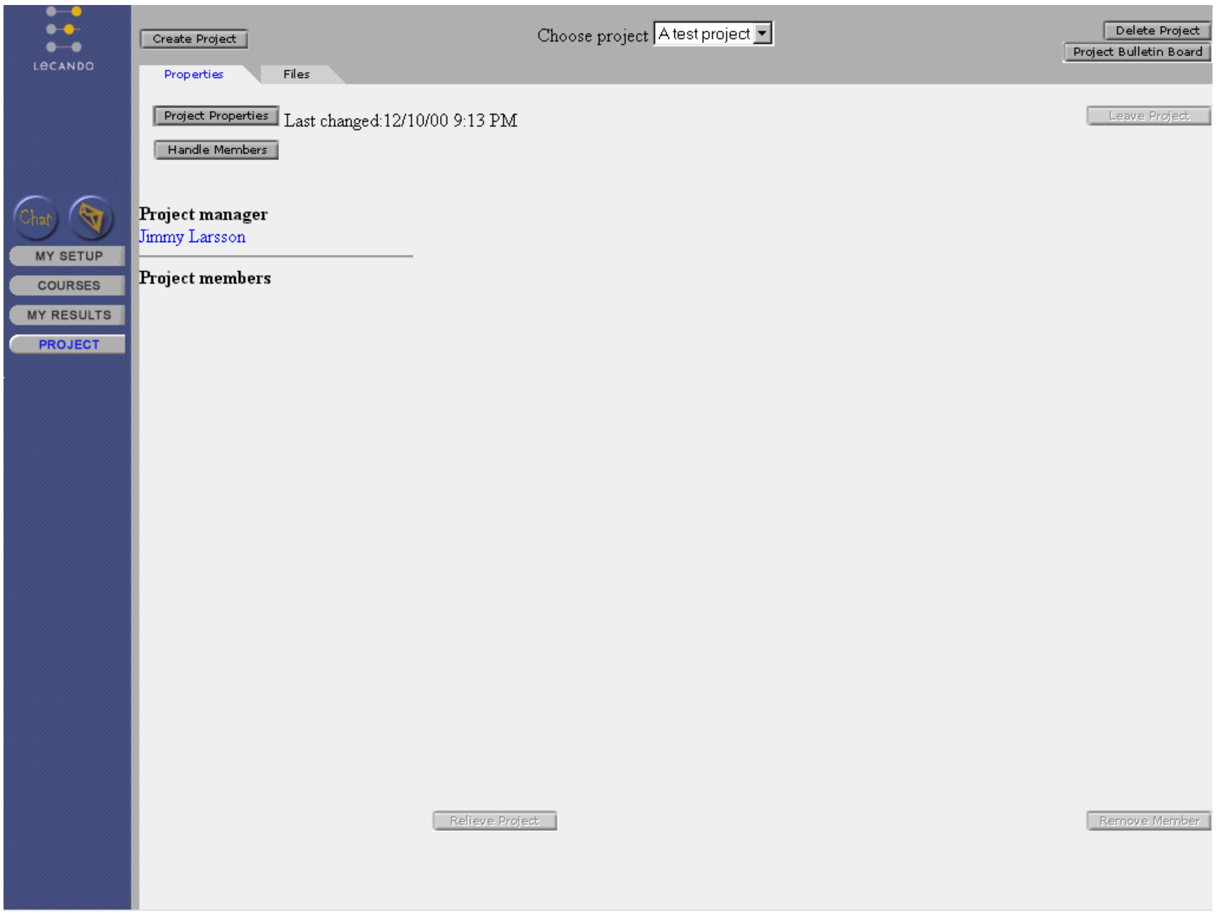
- Creation of a project named “A Test project” in Figure 8.1
- The project overview page of “A Test project” in Figure 8.2
- Adding of members to the “A Test project” in Figure 8.3
- User properties of members can be displayed as shown in Figure 8.4
- A typical directory structure in the filemanager is shown in Figure 8.5
- Figure 8.6 shows the interface when a member creates a directory
- Uploading of a file is shown in Figure 8.7
- Selection of an uploaded file is shown in Figure 8.8
- Locking of a selected file is displayed in Figure 8.9
- Figure 8.10 shows the popup window for properties of a selected file
- The popup window for downloading of a file is shown in Figure 8.11
- The popup window for the news reader is shown in Figure 8.12



Create new project

Project name	A test project
Description	This is a test project
File locking is active	☑
Version handling is active	☑
Quota	50 Mb

Figure 8.1 A project named "A Test project" is being created via a popup window.

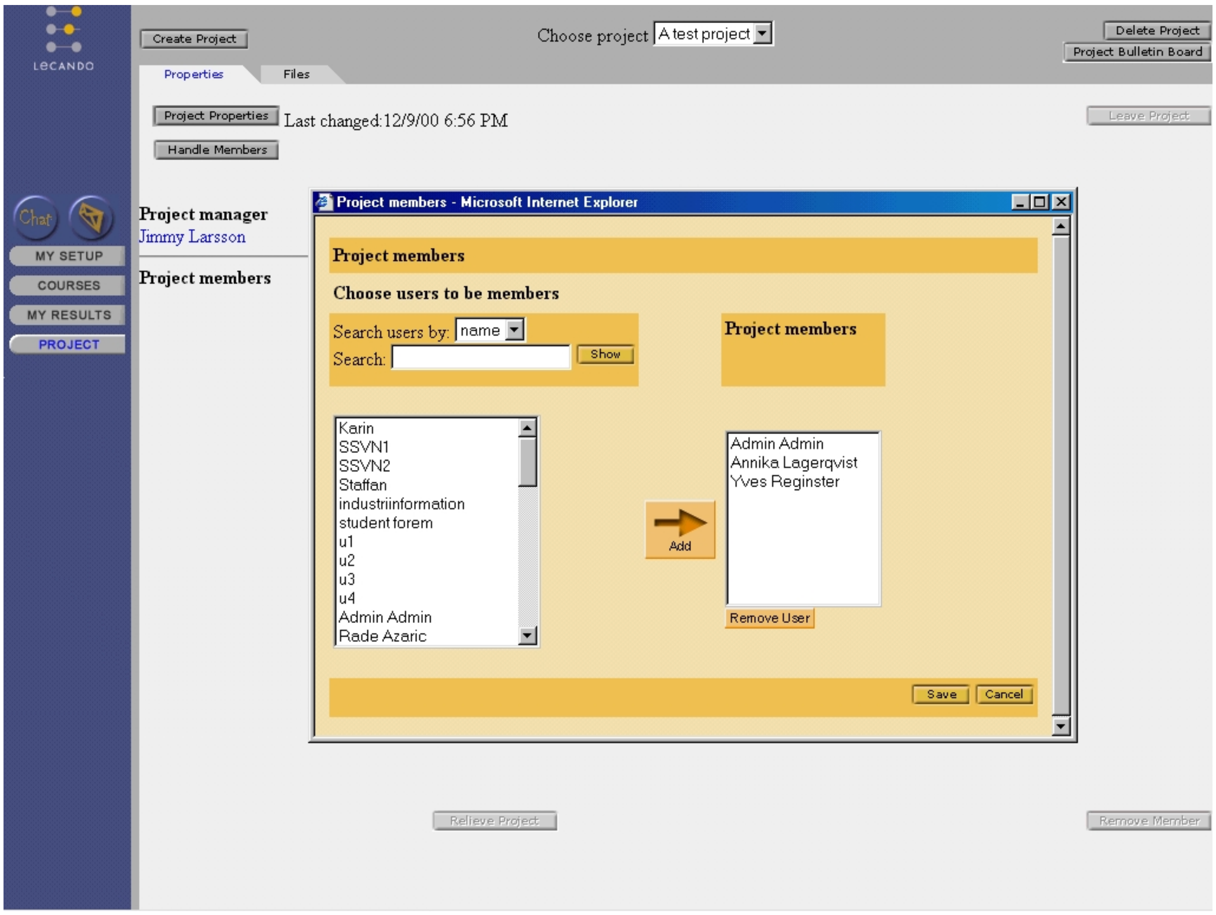


Last changed:12/10/00 9:13 PM

Project manager
Jimmy Larsson

Project members

Figure 8.2 The overview page of the newly created project "A Test project"



Last changed:12/9/00 6:56 PM

Project manager
Jimmy Larsson

Project members

Project members

Choose users to be members

Search users by: name

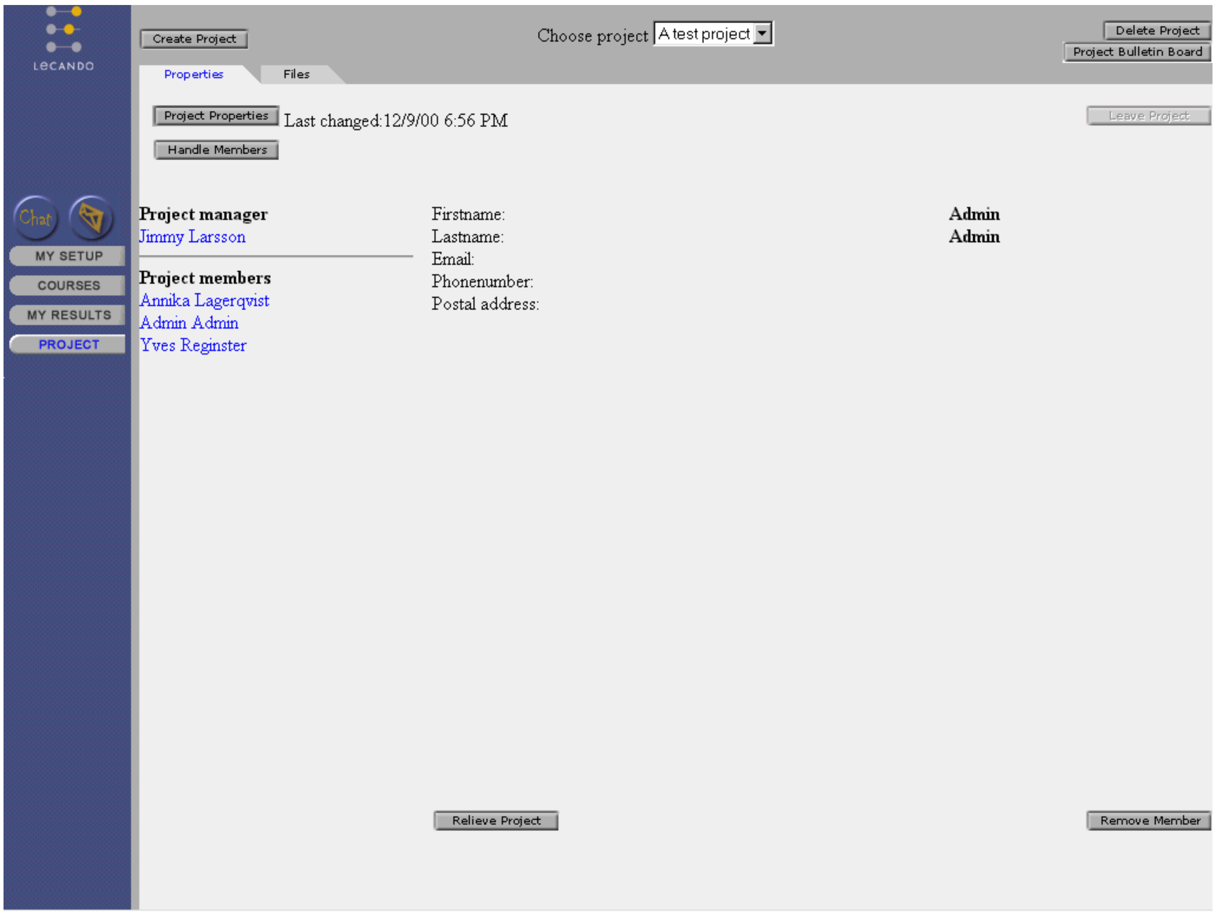
Search:

Karin
SSVN1
SSVN2
Staffan
industriinformation
student forem
u1
u2
u3
u4
Admin Admin
Rade Azaric

Project members

Admin Admin
Annika Lagerqvist
Yves Reginster

Figure 8.3 Three users have been added as members of the project "A Test project"



Last changed:12/9/00 6:56 PM

Project manager
Jimmy Larsson

Project members
Annika Lagerqvist
Admin Admin
Yves Reginster

Firstname:	**Admin**
Lastname:	**Admin**
Email:	
Phonenumber:	
Postal address:	

Figure 8.4 User information is presented after clicking on the member in the memberlist.

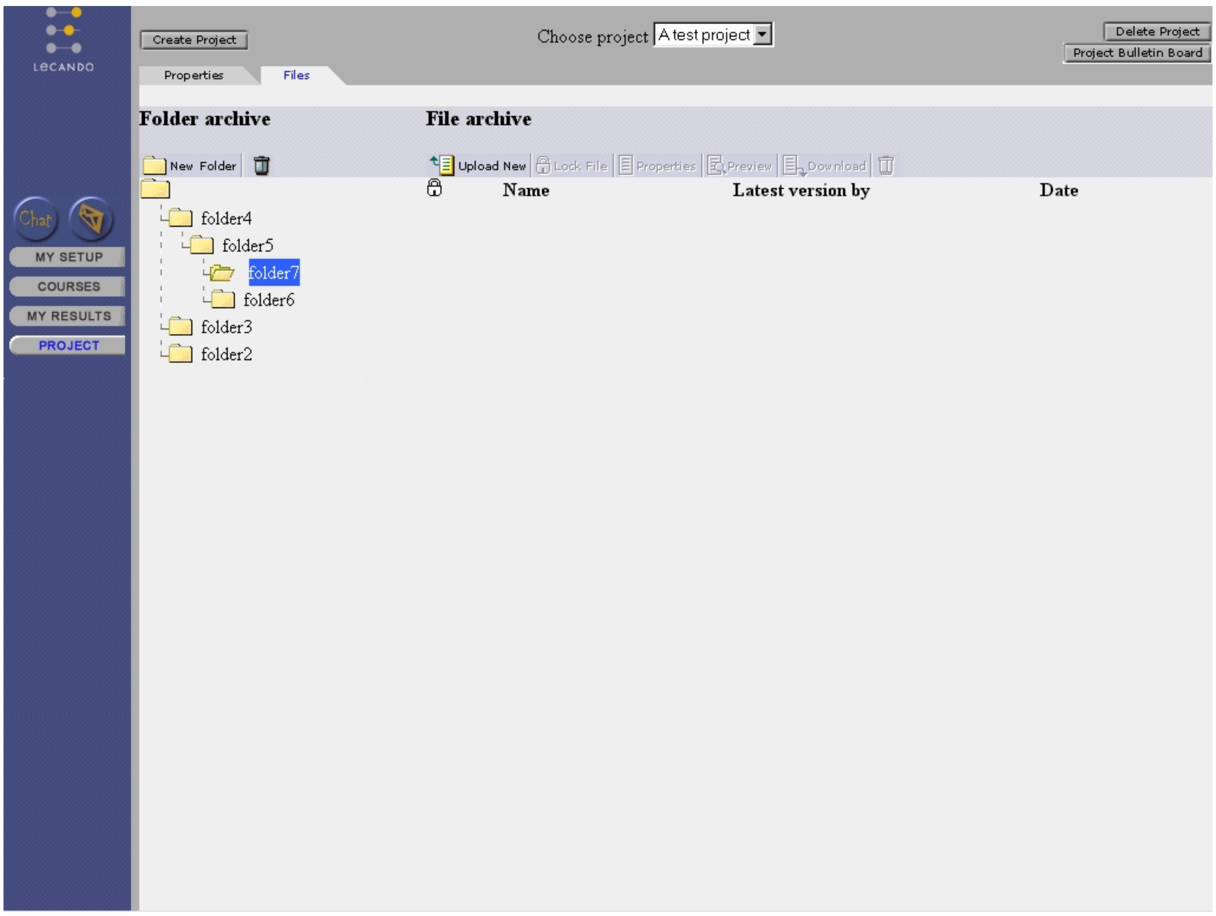


Figure 8.5 A typical directory structure in the file manager. The directory "folder7" is selected and empty.

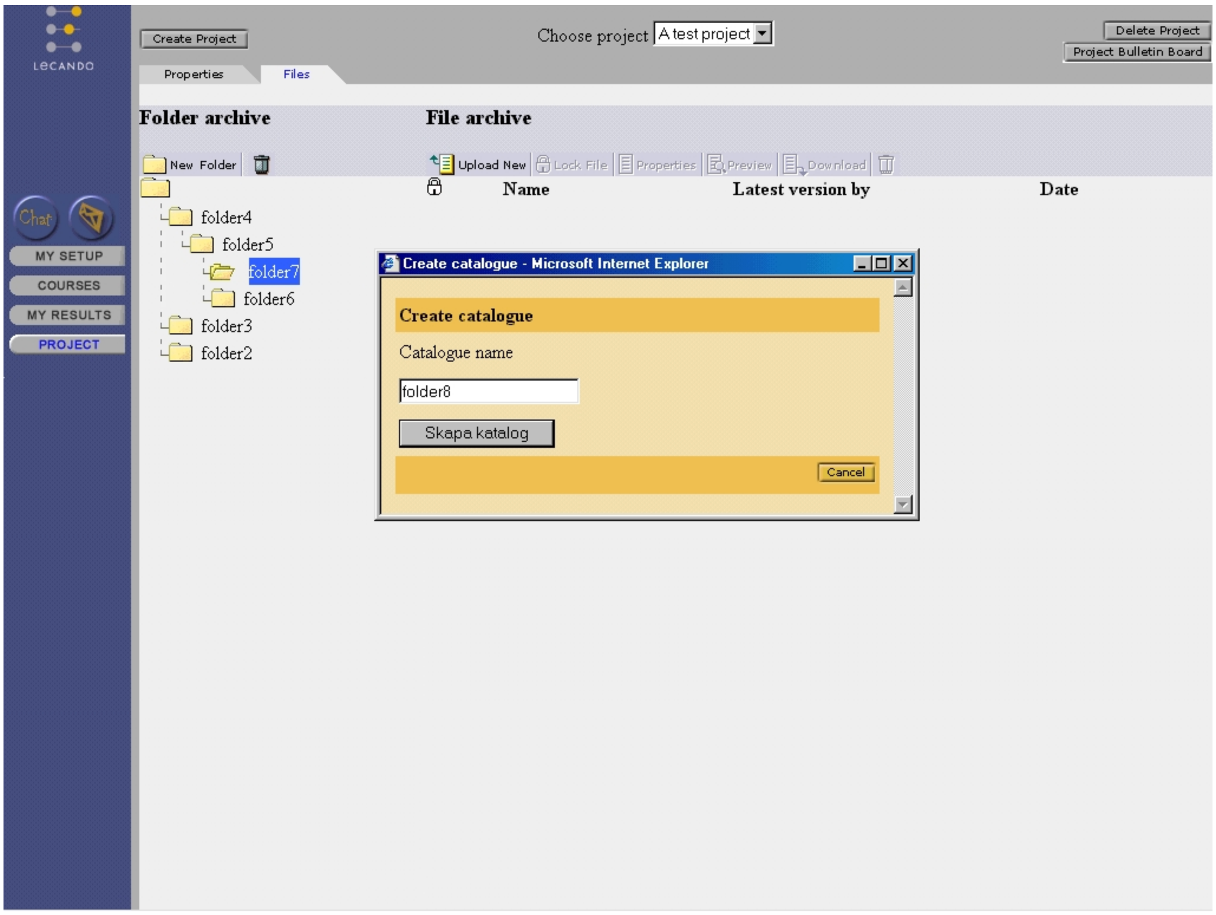
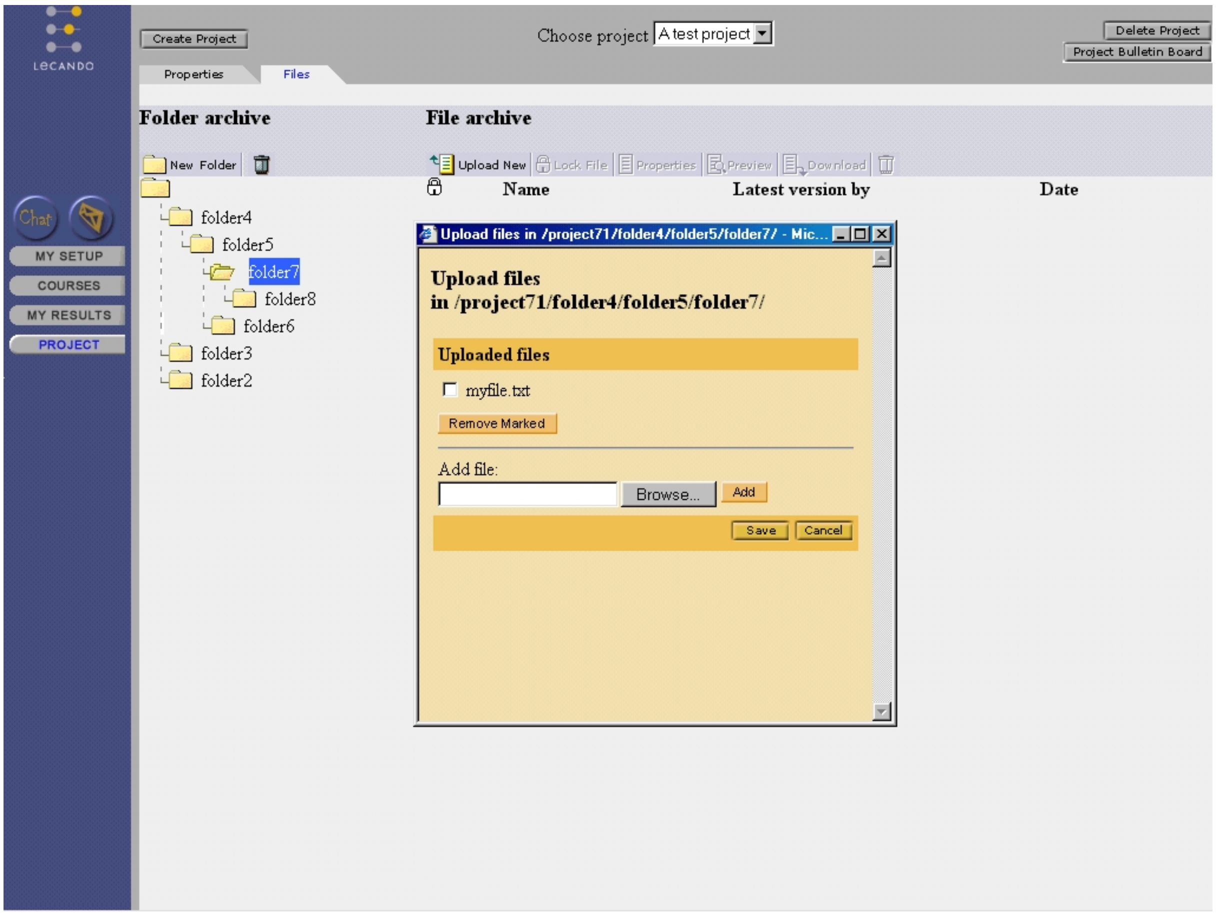


Figure 8.6 A project member creates a new directory named "folder8" in the selected directory "folder7".



Upload files
in /project71/folder4/folder5/folder7/

Uploaded files

☐ myfile.txt

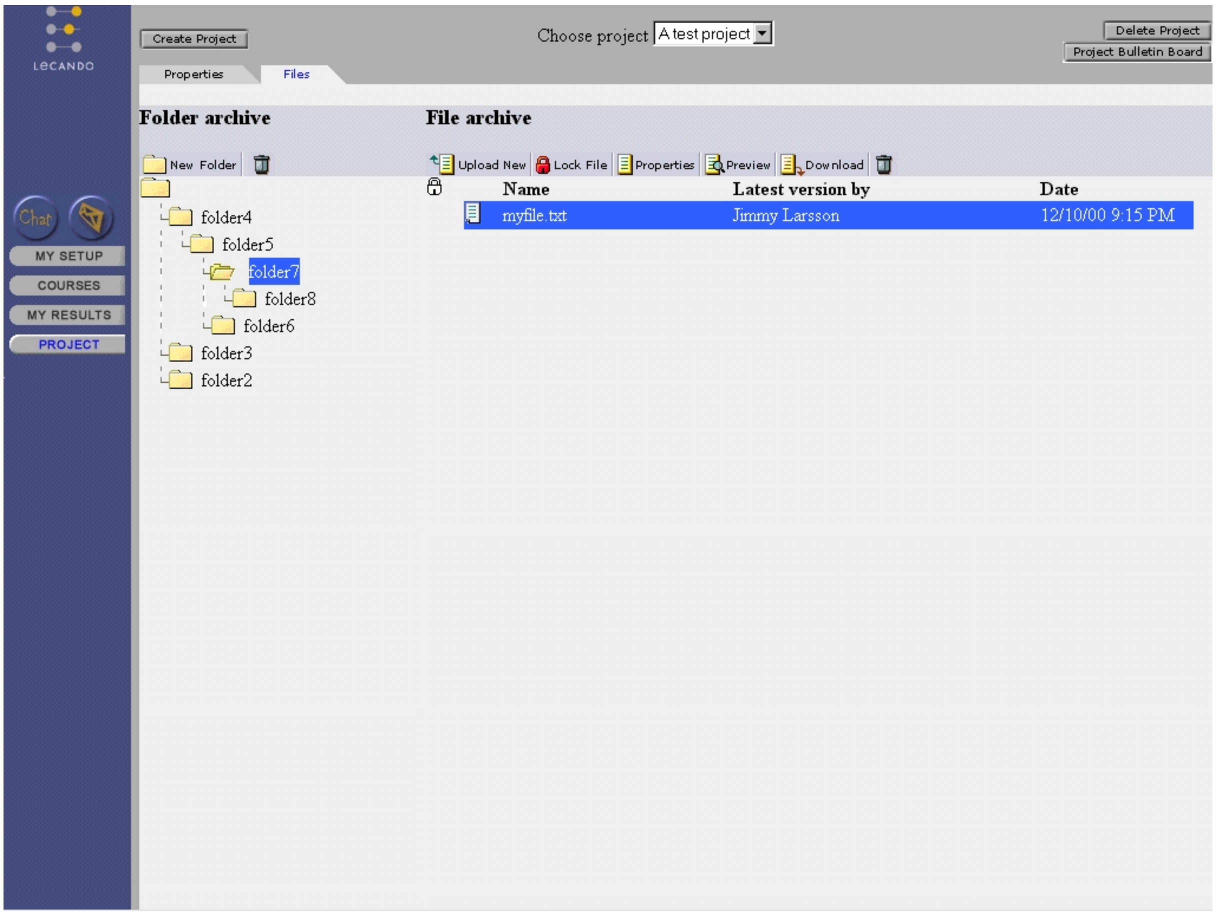
Remove Marked

Add file:

Browse... Add

Save Cancel

Figure 8.7 A member has uploaded the file "myfile.txt". If the member clicks "Save" the file will be stored in the selected directory "folder 7". If the member clicks "Cancel" the uploaded file will not be stored permanently in the project.



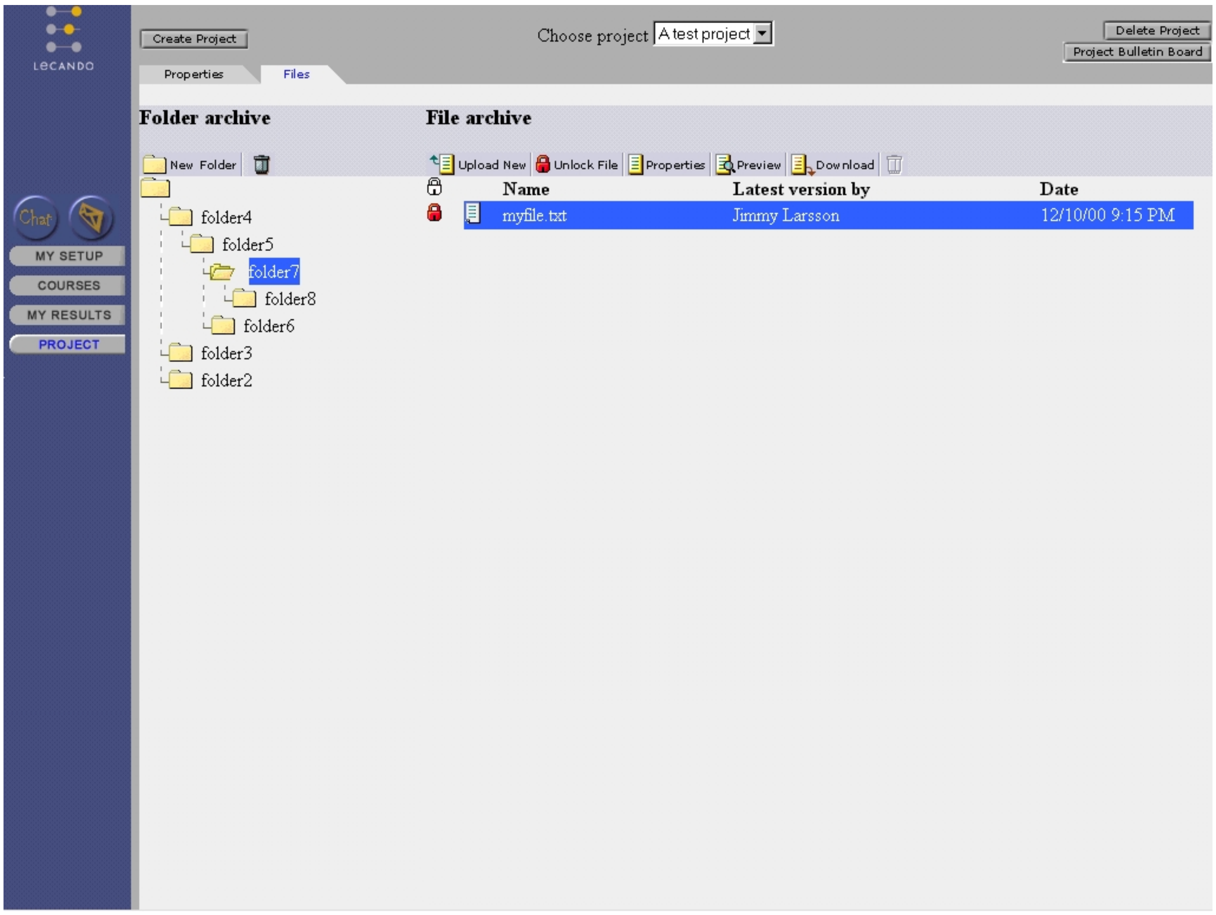
Folder archive

```
folder4
  folder5
    folder7
      folder8
    folder6
folder3
folder2
```

File archive

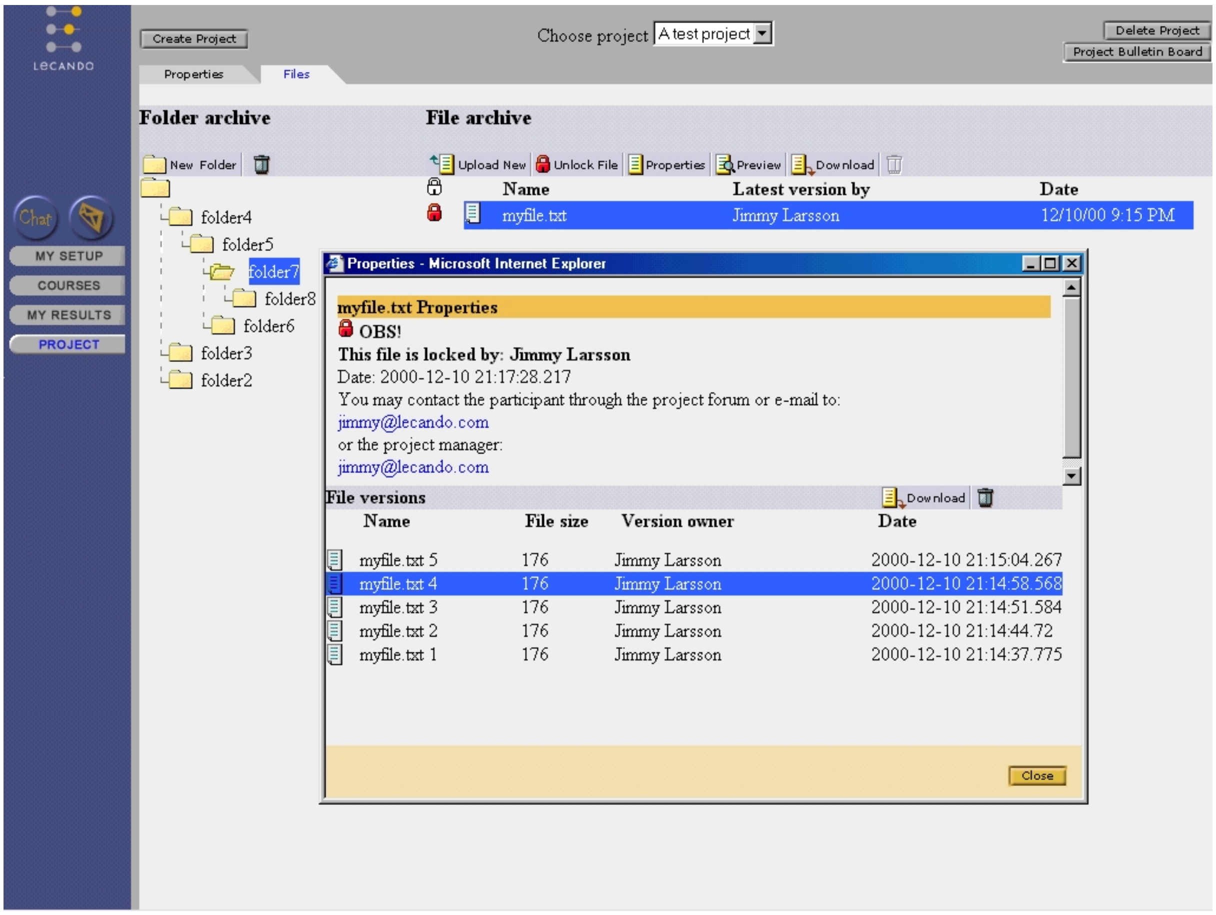
Name	Latest version by	Date
myfile.txt	Jimmy Larsson	12/10/00 9:15 PM

Figure 8.8 A member has selected the file "myfile.txt". The toolbar for the file archive is now active, giving the member options to lock the file, download it, view its contents in a window, examine the properties of the file or delete it.



Name	Latest version by	Date
myfile.txt	Jimmy Larsson	12/10/00 9:15 PM

Figure 8.9 A member has locked the file "myfile.txt". Notice that the trashcan is deactivated and that the button "Lock File" has changed to "Unlock File".



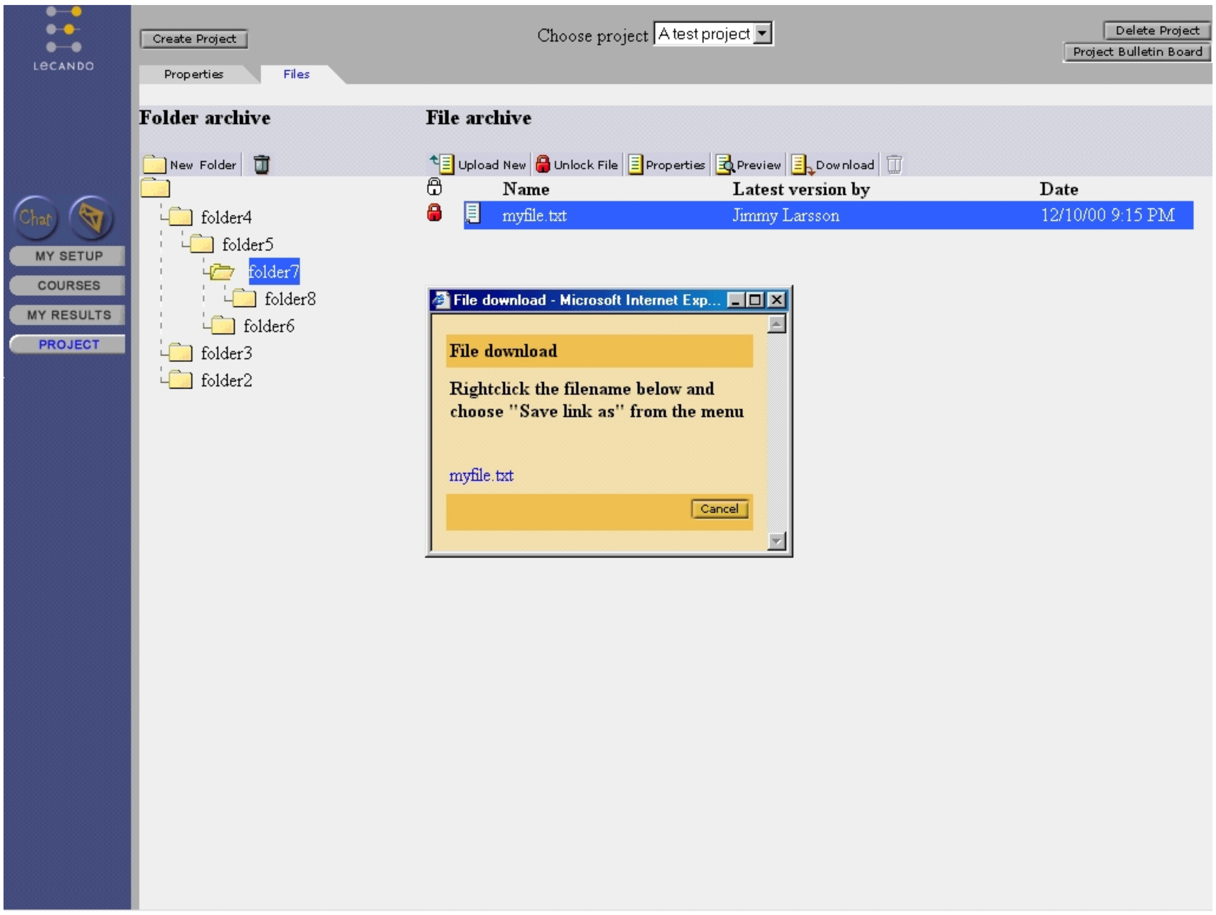
	Name	Latest version by	Date
🔒	myfile.txt	Jimmy Larsson	12/10/00 9:15 PM

myfile.txt Properties
OBS!
This file is locked by: Jimmy Larsson
Date: 2000-12-10 21:17:28.217
You may contact the participant through the project forum or e-mail to:
jimmy@lecando.com
or the project manager:
jimmy@lecando.com

File versions

Name	File size	Version owner	Date
myfile.txt 5	176	Jimmy Larsson	2000-12-10 21:15:04.267
myfile.txt 4	176	Jimmy Larsson	2000-12-10 21:14:58.568
myfile.txt 3	176	Jimmy Larsson	2000-12-10 21:14:51.584
myfile.txt 2	176	Jimmy Larsson	2000-12-10 21:14:44.72
myfile.txt 1	176	Jimmy Larsson	2000-12-10 21:14:37.775

Figure 8:10 A member has pressed the "Properties" button with the file "myfile.txt" selected. A popup window displays all previously uploaded versions of the file. The version number is displayed after the filename. Version number 4 of the file is selected and can be deleted or downloaded via buttons.



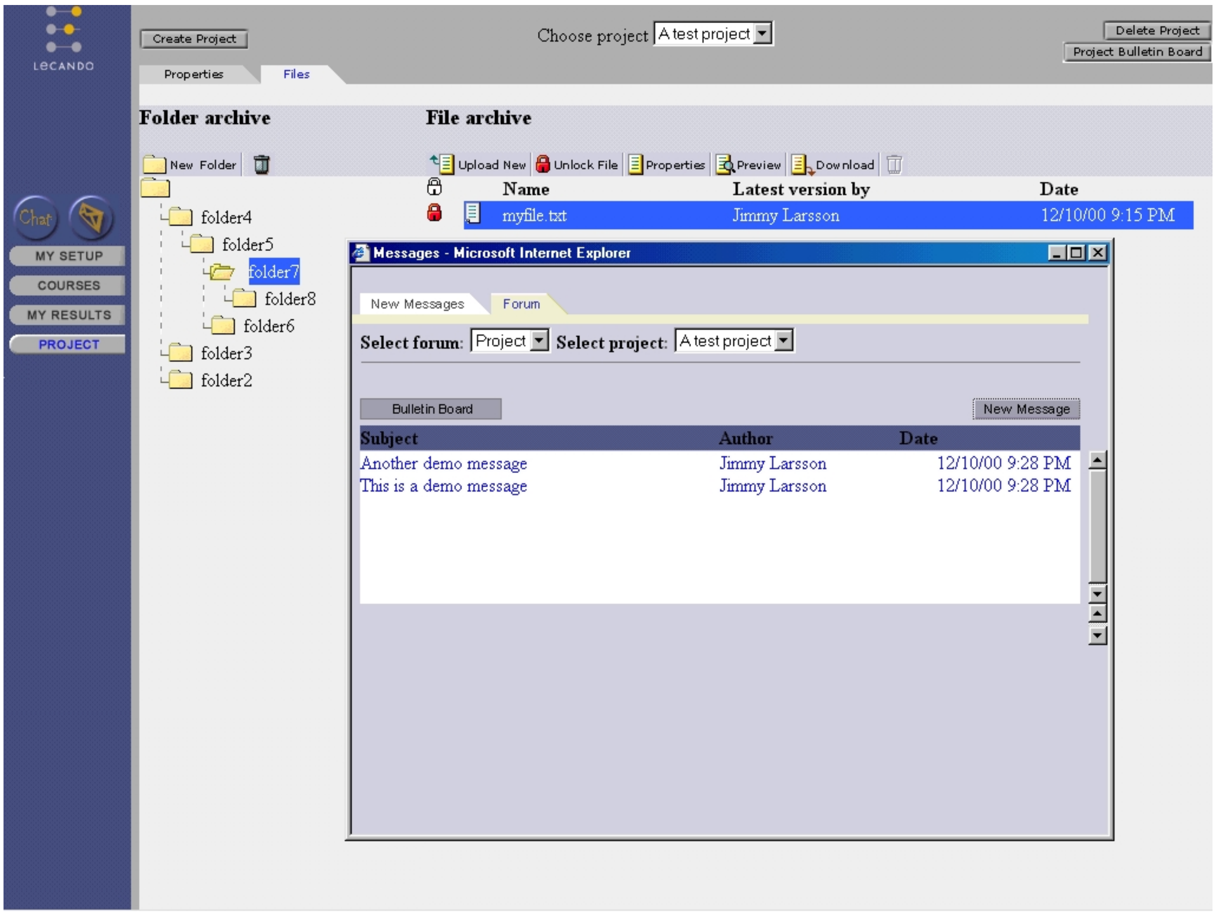
Name	Latest version by	Date
myfile.txt	Jimmy Larsson	12/10/00 9:15 PM

File download

Rightclick the filename below and choose "Save link as" from the menu

myfile.txt

Figure 8.11 The file "myfile.txt" is selected and the download button has been pressed. In the popup window the file can be downloaded by right-clicking on the filename and saving the file.



Name	Latest version by	Date
myfile.txt	Jimmy Larsson	12/10/00 9:15 PM

Select forum: Project Select project: A test project

Subject	Author	Date
Another demo message	Jimmy Larsson	12/10/00 9:28 PM
This is a demo message	Jimmy Larsson	12/10/00 9:28 PM

Figure 8.12 The button "Project Bulletin Board" has been clicked. The news reader is displayed in a popup window and all the messages for the project "A Test project" are listed. By clicking on the tab "New Messages" all messages the user has not read will be listed.