ISSN 0280-5316
ISRN LUTFD2/TFRT-5791--SE

New Web based HMI portal for Tetra Pak equipment

Johanna Byrlind
Kajsa Karlsson

Department of Automatic Control
Lund University
February 2007

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
February 2007

Document Number
ISRNLUTFD2//TFRT--5791--SE

Author(s)
Johanna Byrlind and Kajsa Karlsson

Supervisor
Mattias Wallinius at Tetra Pak D&E in Lund.
Karl-Erik Årzén at Automatic Control in Lund, (examinator)

Sponsoring organization

Title and subtitle
New Web based HMI portal for Tetra Pak equipment. (Ny webbaserad HMI portal för Tetra Paks utrustning)

Abstract

The topic of this master thesis is the development of a new Web based HMI portal for Tetra Aptiva Aseptic. The aim is an open source code project where the basic idea is examined and partly implemented during this master thesis. Through a new technique, developed by Tetra Pak, clients can subscribe on topics of interest, which can be viewed on any trusted and connected computer within the Tetra Pak network.
The development of a Web HMI includes an investigation about the possibilities to separate the front and the back-end programming. To put the separation into practise JavaServer Pages' rich architecture of Custom Tag Libraries was used. A few tags were developed and connected to realistic scripts to get an indication about the technique's possibilities.
The result showed that this approach has many hopes for the future. There was no problem separating the front and back-end programming, which will benefit software development projects at Tetra Pak in the future. People with different skills can come together and contribute with their knowledge from their domain of expertise. However, some difficulties occurred while testing this application.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
72

Recipient's notes

Security classification

The report may be ordered from the Department of Automatic Control or borrowed through:University Library, Box 3, SE-221 00 Lund, Sweden Fax +46 46 222 42 43

Preface

This thesis marks the end of Johanna Byrlind's and Kajsa Karlsson's Master of Science degree in Information and Communication Engineering. The thesis is based on the work at the Development & Engineering division at Tetra Pak in Lund, Sweden.
The time at Tetra Pak D&E has not only taught us software development, but also a great deal of social behaviour, politics and how larger organizations work.

We would like to thank Mattias Wallinius, our supervisor at Tetra Pak D&E and Professor Karl-Erik Årzen, our examiner at Lund University. We would also like to thank our co-workers at Tetra Pak, family and friends who have been a great support during this master thesis.

Table of content

Glossary

Below is a list where abbreviations and other ambiguities are explained.

- **DTD**
  DTD is an abbreviation for Document Type Definition and is a way to describe the structure of an XML document.

- **CA**
  A Certificate Authority certifies that the certificate holder is who they say they are.

- **CM**
  Configuration Management

- **Design Patterns**
  Used within object-oriented programming to enhance code reusability and support programmers with solutions for common problems. [1]

- **HMI**
  The aggregate by which people interact with machines is called Human-Machine interface, HMI. The user can manipulate the system via input while the system produces outputs according to the input.

- **FDA certified**
  U. S. Food and Drug Administration have certified it.

- **J2EE**
  Java Platform, Enterprise Edition (Java EE, also called J2EE) is the industry standard for developing portable, robust, scalable and secure server-side Java applications.

- **JDK**
  The Java Development Kit is a Sun product aimed at Java developers.

- **MVC**
  MVC is an abbreviation for Model-View-Controller. It is a design pattern used within software engineering.

- **Open source code**
  Open source code is free to use and read and is available for modification and distribution for everyone interested. Modifications made by volunteers are often offered back to the originator, whom can choose to make it a part of the official version.

- **OpenSSL**
  OpenSSL is an open source implementation of the SSL and TLS protocols.

- **PLC**
  Programmable Logic Controller, PLC, or Programmable Controller is a small computer used for automation of industrial processes.

- **RRL**
  Stands for Remote Reference Layer and is a part of Java's Remote Method Invocation protocol.

- **SSL**
  SSL stands for Secure Sockets Layer and is a way to create a secure HTTP connection between a client and a server.

- **TLS**
  TLS stands for Transport Layer Security and is successor to SSL. There are a few differences between the latest version of SSL and the TLS protocol, but the main structure remains.

- **URI**
  The purpose of a Uniform Resource Identifier is to identify or name a resource. It is basically a compact string of characters which are defined in schemes defining a specific syntax and associated protocols.

- **W3C**
  The World Wide Web Consortium, W3C is an organisation for standardization on the web. They develop interoperable technologies, like specifications, guidelines, software, and tools to lead the Web to its full potential. [2]

1 Introduction

A modern HMI for a Tetra Pak filling machine is today mainly a tool to draw pictures and bind them to data from a control system. It then normally has a runtime that renders the pictures and handles some script code to animate and add some functionality to the HMI. The configuration is then stored in proprietary format in either flat files or configuration databases. Some extra functionality is also required as for example networking i.e. the solution should be viewable at different places. Data connectivity to get information from the system is also required.

If one looks at the requirements one finds that a web application is doing most of the things mentioned above, except the animation part, when connected to data from the control system. There have been several attempts to make HMIs viewable on browsers and most of them make use of insecure technology such as ActiveX to bypass the problems with redrawing web pages. At Tetra Pak Development & Engineering there is a solution to this and that is to utilize the scripting engine of the browser. This is done with a simple push technology where scripts are being pushed to the browser and executed there. This circumvents the shortcomings of the browser and emphasises the advantages such as open formats and standards. The new SVG standard also handles the problem of scalable vector graphics.

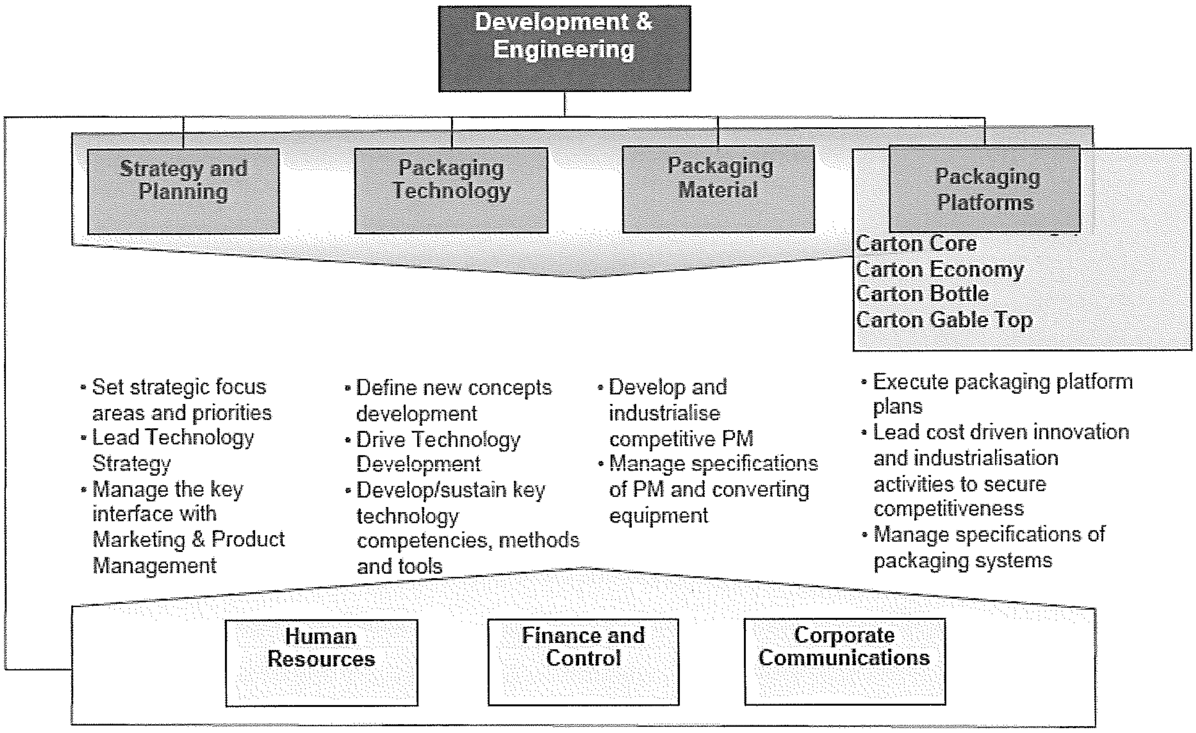
There are several advantages for Tetra Pak. To mention a few: a cut in licence costs, becoming supplier independent and more flexibility in the way to do business. All are strong arguments, which can not be neglected if striving for market leadership.

2 Brief About Tetra Pak

Tetra Pak is one of the largest providers of processing and packaging solutions for food and drinks with more than 20 000 employees. They are committed to making food safe and their packages are sold in more than 165 countries and in more than hundreds of different types of carton packaging formats. They develop state-of-the-art processing solutions and design and service complete plants. That means that they provide the customers with packaging machines, packaging material and distribution systems.

This master thesis was performed in the Development & Engineering division. Their vision is to regain technological leadership, use cost-driven innovation to deliver more for less and to create winning solutions to achieve sustainable profitable growth. "Development & Engineering takes identified needs from the market and turns them into products and maintains them over their life cycle."

The organisation is naturally formed around these areas of expertise: Strategy and planning, Packaging technology, Packaging material and Packaging platforms. In Figure 2.1 a schematic picture of the Tetra Pak D&E division is shown.



Development & Engineering
Strategy and Planning
Packaging Technology
Packaging Material
Packaging Platforms
Carton Core
Carton Economy
Carton Bottle
Carton Gable Top
• Set strategic focus areas and priorities
• Lead Technology Strategy
• Manage the key interface with Marketing & Product Management
• Define new concepts development
• Drive Technology Development
• Develop/sustain key technology competencies, methods and tools
• Develop and industrialise competitive PM
• Manage specifications of PM and converting equipment
• Execute packaging platform plans
• Lead cost driven innovation and industrialisation activities to secure competitiveness
• Manage specifications of packaging systems
Human Resources
Finance and Control
Corporate Communications

Figure 2.1The Development & Engineering division of Tetra Pak

More information and updated numbers can be found at http://www.tetrapak.com/ (public) and http://development-engineering.tetrapak.com/Mainframe.aspx (internal).

3 Requirements, Goals and Limitations

3.1 Web development tools

Automation & Line integration today models the control software with UML. To develop web applications there were three existing technologies; PHP, .Net and Java to choose from. .Net is used to develop applications for the web, but since the aim for this master thesis is to develop without proprietary formats, the use of .Net was unfeasible. PHP is a widely-used general-purpose scripting language that is suited for Web development and can be embedded into HTML.[3]

Java was chosen since it is open source and Sun is now the largest contributor to the open source community. [4] Java has the Apache Tomcat JSP platform for web development in Java. Due to our prior knowledge in Java we were able to focus more on the problems to solve than the syntaxes.

One also needs to use the standard web formats as XHTML, SVG and JavaScript. On the wish list is also to separate the coding from the graphical design work when developing. This is exactly what the Tomcat platform can offer you in form of tag libraries. Here much work already has been done to be used in form of JSP Standard Tag Library. Still techniques need to be evaluated and selected.

3.2 Data binding

It is vital that the data binding process is supported. This is when behaviour and objects in the web application are being coupled to the data from the control system. The web tool must allow plug-in extension development for such functionality.

3.3 Performance

The total system performance must be in line with the requirements on Tetra Pak filling machine HMIs. Our aim is to develop an application that is so fast that the human eye can not detect any delay.

3.4 Security

There must be a solution which can handle the security aspects of an HMI solution. For example the HMI control point must be set so that the automat only can be controlled from one point at a time. Also different actors on the system must have different functionality available to them. It is beneficial to gather all access points into one access point, i.e. the web server. Then it is much easier to analyse the loopholes and address them appropriately.

3.5 Limitations

In this project there is neither consideration taken to how the PLCs is constructed nor how the communication with the HMI is conducted. In this rapport there will be no evaluation of the existing HMI. The focus for this project is on the technical side on the HMI rather than the usability aspect.

4 Software Engineering Methodologies

Our objective was to develop software in a structural way with different software development methodologies in mind.

4.1 Rational Unified Process

This thesis is in some ways influenced by the method Rational Unified Process. RUP takes an evolutionary approach to development.[5]

The method of RUP was studied first and the parts were chosen that could apply to our work.

A RUP lifecycle has four phases:

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Each phase ends with a well defined milestone. If the project does not pass this milestone it can either be cancelled or it can repeat the phase and hopefully better meet the specified goal the next time. [6]

In the following sections a brief description about the RUP-phases can be found.

4.1.1 The Inception Phase

The aim in this phase is to get stakeholders coherence on scope definition and cost/schedule estimates. In the inception phase a high-level requirements model is made. The requirements are used in several phases, both for designing a solution, coding and testing the software. All activities relate back to the requirements. Also a high-level plan for how the project will proceed is completed, that is a project plan, an initial risk assessment and a project description to help the project achieving the best quality possible. [7, 8]

4.1.2 Elaboration phase

One of the most important tasks in this phase is to define how the problem is to be solved. The software architecture is also decided, it must be stable and sufficient to satisfy the requirements. To prove the architecture it is recommended to implement and test a skeleton of working code. [9, 10]

4.1.3 The Construction Phase

The focus of the Construction phase is to develop the system to the point where it is ready for deployment. The bulk of the coding takes place in this phase. At this point in the software engineering process the requirements need to be prioritized to get satisfying software. Early releases of the system can be deployed, either internally or externally, to obtain user feedback [11]. In larger projects, several construction iterations may be developed.[12]

4.1.4 The Transition Phase

The Transition phase focuses on delivering the system into production. There will be testing by both system testers and end-users to validate it against the end users' expectations. The transition phase also includes training of end users, support personnel and operations staff.

4.2 Agile

There is a trend in the software industry now to move towards working in an agile way. Already in November 2005 14 % of North American and European enterprises claimed that they were using Agile processes and 19% were interested in Agile and planned to deploy it that year. [13]

Agile is a name for several different methods like XP, Crystal and SCRUM. It is based on the Manifesto for Agile Software Development[14]. The main points for the manifesto are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Most of the agile methods consist of working in short iterations to minimize the risk of different threats to the project. In Agile methods face-to-face is the preferred way of communication, over written documents.[15]

Of the different Agile methods the one we found most useful was SCRUM [16], yet there are parts that had to be modified to suit our work more appropriately.

SCRUM Statement- In SCRUM there is a Scrum-meeting every day. The purpose of this brief meeting is to eliminate all speed impediments for the group. The questions to be answered at each meeting are:

- What have you done since the last meeting?
- What will you do between now and the next meeting?
- Is there anything preventing you from doing what you have planned?

The daily meetings were never officially part of our work routine. But due to the fact that this master thesis is performed by two people it came natural that the immediate work was organised on a daily basis.

SCRUM Statement- One important part of SCRUM is the rather short iterations of 30 days. The iteration is sometimes called a sprint and has fixed goals for the group to achieve.

It was good in this project to have a definite goal to work towards. The iterations were not always 30 days; sometimes they were longer and other times considerably shorter. However, it was always our ambition to develop software that was both compliant and executable in all iterations. In most cases the goal was met and a small demonstration could take place.

SCRUM Statement- Self organizing teams are an essential part of SCRUM. Even if there are people more trained in certain areas the work should be self organized.

Since the thesis is only performed by two people and there are many more roles to fill, it was natural for us to do more than one role. There was never a fixed division of labour; both did the architectural design, the coding and the testing.

4.3 CM

Configuration Management is the process of identifying and defining the items in the system, controlling the change of these items throughout their life cycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items.[17] One way to make sure that this occurs is to use a version control system. All parts of this thesis were possible to add to the version control. In our case we used Subversion. Its aim is to be a compelling replacement for CVS in the open source community. The software is released under an Apache/BSD-style open source license. [18]

4.4 Conclusion and Evaluation of the Engineering Methodologies

It was extremely useful to use an engineering framework for this master thesis. Especially since our experience of software development for commercial use was limited. One of the major advantages with using a framework was that the work grew to be much more organised and structured. When the different methodologies were studied we found that many of them were quite similar, and that they all aimed to increase the quality of the software.

At the start of this master thesis there was an initial meeting with both Mattias Wallinius (Advisor, Tetra Pak) and Karl-Erik Årzen (Examiner, LTH). There was a discussion about the scope of the project and together we made a project plan with rough time estimates.

The first four weeks in the elaboration phases were used to fully grasp the problem to solve. Furthermore we found it vital to understand the context in which the HMI was going to be used. To be able to decide what architecture to use there was an in-depth evaluation of the different techniques available. After deciding which techniques to use several practical coding examples were made to determine that the different techniques worked sufficiently well together. Since a few coding examples already were conducted it was natural to use them as a base for further coding during the construction phase.

There have also been weekly meetings with our supervisor, Mattias Wallinius, with feedback to monitor the progress and to assure the quality of the project. In this project there was no requirement elicitation, but the different requirements were set from the beginning of the project. No formal requirement prioritisation has been done, but during the meetings the supervisor has structured our work the best possible way for Tetra Pak, so in a way the prioritisation has been continuous.

There have also been a few early releases to our advisor, but not any external releases to get feedback. The vision is to present a platform and continue developing the application as an open source project.[19, 20]

5 Overview of our solution

The possibility to separate the content model from the presentation has brought about new opportunities for software developers. Programmers can concentrate on writing code and Web page designers do not need to know anything about the logic and functionality behind. This work procedure results in a well-presented and solid product, where the groups contribute with the part they master the best. This has many advantages, but one can not forget the challenge to merge these two parts together. It will include good communication and a good understanding about the different set of concerns that can arise.

The solution can be built upon two different approaches. One is based on a Servlet, TetraPush, while the other is using an Applet, Push Applet. The Applet solution was created because one thought that the performance of the Servlet solution would not be sufficient. However, tests showed that it was performing well enough and this together with the fact that it could go through a proxy resulted in that it became the solution chosen for further development. In Chapter 9.1 one can find more information about the Servlet solution, while further reading about the Applet can be found in Appendix G.

Object-oriented development's main goal is to let each object do its own work independently. To make this possible design patterns are used to support this distribution of responsibility. In this master thesis application intelligence is placed in Servlets and beans while JSP pages retrieve content when rendering. Our solution is based on a MVC pattern but also include the Observer and Mediator pattern. This is due to the specific purpose of the project where the Observer is intended for notification and Mediator for handling the communication between the different objects within the Tetra Push solution.

One feature of JSP v 1.1 that aids the separation between logic and presentation is the custom tag library. This technology supports the development of reusable modules which is invoked by using custom tags in a JSP page. This collection of custom tags forms the tag library. When designing a tag library the author must think about the usability so that the tags facilitate the work for the front-end page designer. Designing with usability principles in mind, one can increase the chances that the web page designer understands and can use the library's functionality in her work.

In the TetraPush system one or more clients are connected to the server. The clients access the server by sequences of service requests called a session. A master design was developed for controlling the client's rights to manipulating data in the web HMI. When implementing the master design there was a requirement to be able to maintain information in the session. This is called session state and was used for controlling the possessor of the master role. Only one client can possess the master role at the same time while others can send a request for master. The possessor can decide to release it directly or keep it. In case of inactivity the master role will be released automatically.

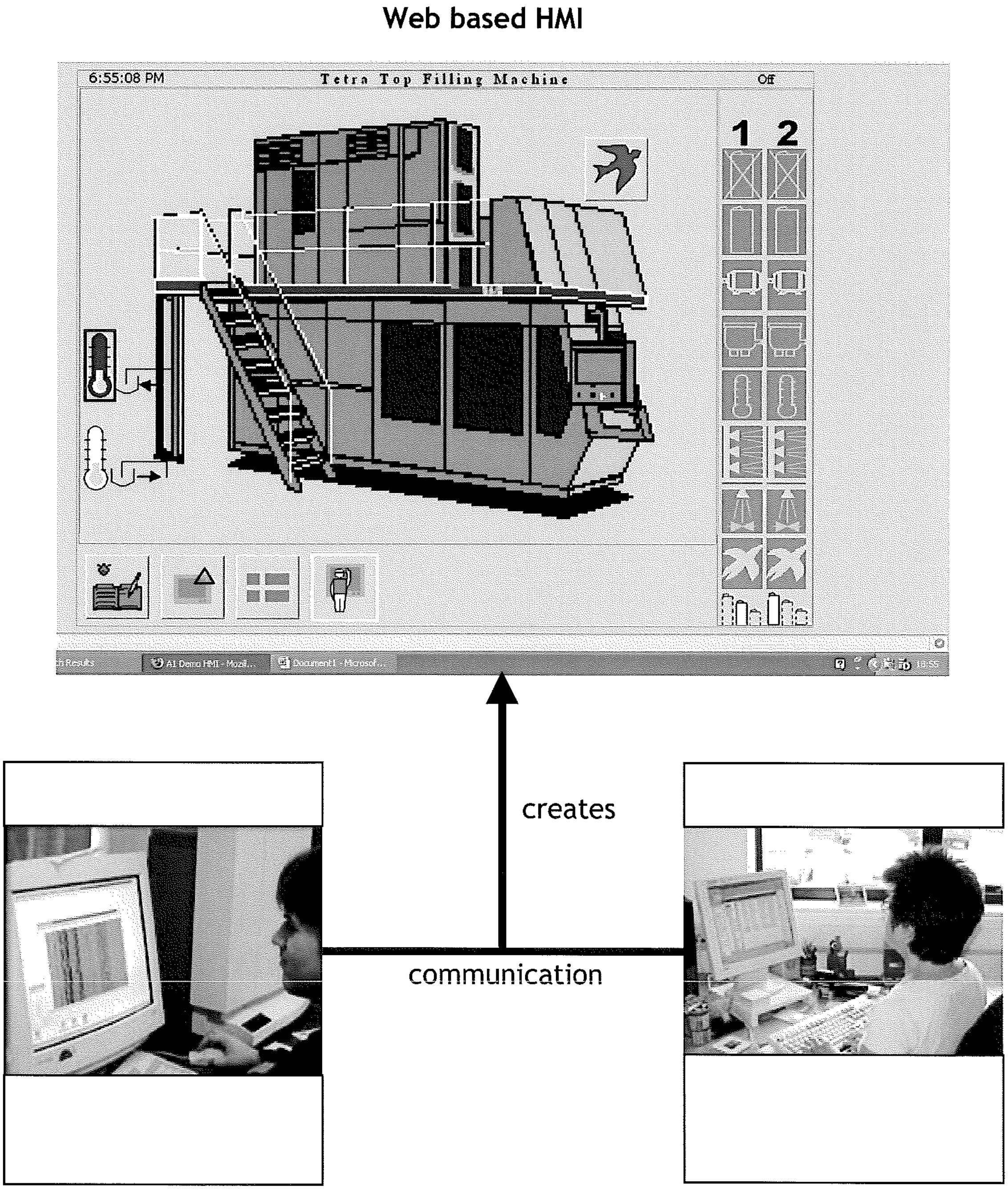


Figure 5.1 An overview showing how merging two different knowledges can improve software development and design.

6 Techniques

During the elaboration phase several techniques were evaluated to find the most suitable ones for this project. When doing this consideration was taken to the way the new web is developing. After years battling about browser standards market leaders have been able to establish a web technology standard and future development adapts to its guidelines. To mention some of the technical standards: HTML, CSS, XML and JavaScript. Old standards are not compatible with new ones and to create such an adaptation will skyrocket the costs. This has brought out new ways of doing things.[21] To read more about Web 2.0 see Appendix I.

In this section an explanation of each chosen technique will follow and why they were selected.

6.1 Client-side techniques

Client-side techniques are those that are performed by the client in a client-server relationship. In this case the client is a web browser that runs on a user's local computer. Due to the fact that the browser represents a web based HMI a connection to the server will be constant.

6.1.1 HTML

HyperText Markup Language, HTML, is a mark-up language that is used for creating web pages. An HTML file is a text file containing small mark-up tags**.** The mark-up tags tell the Web browser how to display the page. It can be headings, paragraphs, lists but also interactive forms, embedded images, and other objects. It can also contain embedded scripting language code.

Each version of HTML has attempted to reflect greater consensus among industry players so that their documents will not become unreadable in a short period of time.[22,23]

"HTML 4 extends HTML with mechanisms for style sheets, scripting, frames, embedding objects, improved support for right to left and mixed direction text, richer tables, and enhancements to forms, offering improved accessibility for people with disabilities."[24]

6.1.2 XHTML

Extensible HyperText Markup Language, XHTML, is also a mark-up language but has a stricter syntax. Another difference is that HTML is an application of SGML (Standard Generalized Markup Language) and XHTML is an application of XML. SGML is a very flexible markup language. XML is derived from SGML and is a specific subset of SGML. XML is aimed to be simpler to parse and process than full SGML.[25, 26,27]

6.1.3 Cascading Style Sheets

CSS is a programming language that is used together with HTML-documents to define how the content of a web page should be presented. Using CSS can not only structure web sites to guarantee consistency, but also add new look-and-feel to them. CSS files include rules how the different elements should be displayed on the page. Below are the three different kinds of style sheets described, starting with the one with highest level of authority.[28]

- Author styles (style information provided by the web page author), in the form of

    - external style sheets, i.e. a separate CSS-file referenced from the document
    - embedded style, blocks of CSS information inside the HTML document itself
    - inline styles, inside the HTML document, style information on a single element, specified using the "style" attribute.
- User style
    - a local CSS-file specified by the user using options in the web browser, and acting as an override, to be applied to all documents.
- User agent style
    - the default style sheet applied by the user agent, e.g. the browser's default presentation of elements.

These three levels override each other depending on their priority hence the name Cascading Style Sheets. CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element.

CSS2.1 is the last recommendation from W3C, dated in January 2003.[29] CSS3 working drafts and candidate recommendations are in progress and for more details see http://www.w3.org/TR/css3-roadmap/.

6.1.4 JavaScript

JavaScript is a scripting language and was designed to add interactivity to HTML pages. It was developed by Netscape in 1995. JavaScript consists of executable computer code and thanks to the functionality of being an interpreted language, JavaScript can execute without preliminary compilation.[30]

In JavaScript objects are created by attaching methods and properties to otherwise empty objects at run time. Some of the things the JavaScript can do are opening a window with a possibility of control, giving additional information to the user when the mouse cursor is moved over the object and validating web forms.[31]

The JavaScript engine is embedded in the host environment where the web browser is the most common. The JavaScript engine is also called the JavaScript interpreter; it interprets the JavaScript source code and executes the script.[32]

The most common variant of JavaScript is the client-side Java Script. The definition comes from that a JavaScript interpreter is embedded in the web browser. Client-side JavaScript combines the scripting ability of the interpreter with the document object model defined by the web browser.[33]

6.1.5 XML

XML is an abbreviation for Extensible Markup Language and recommended by W3C as a general purpose markup language. XML is a way to structure data to make sharing data across different system connected to Internet, possible. This set of rules describes how textual formats shall be constructed. It facilitates a computers work to generate and read data but also guarantees computer structure to be unambiguous. Other features are its platform independency and ability for extension.[34]

6.1.6 DHTML

When creating a dynamic web page several technologies are bound together. There is HTML for static page content, a client-side scripting language such as JavaScript, the advantages of CSS for the presentation and finally the Document Object Model, DOM. This collection of technologies for creating interactive and animated web sites is called Dynamic HTML, DHTML.[35]

DHTML is not a standard recommended by W3C. W3C once said: "Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."[36]

HTML is the core of DHTML, while CSS, DOM and JavaScript contribute to its dynamic property. CSS controls the document's presentation; the DOM organizes the HTML components in hierarchal structure of named objects, while JavaScript creates the effects by accessing the components of the DOM and changing them on the web page. There are several advantages of using DHTML. Comparing to other programs that create for example animations, DHTML files are much smaller. There is no need for any special software like Flash [37], as most browsers already support the techniques involved. The increasing popularity has brought new light on the work on a final standards document for the DOM, and the already decided standards document for CSS.

By including DHTML in software projects web programmers contribute to faster downloads, no need for plug-ins and a development towards a W3C standard. The fact that it is not yet a standard makes it a little bit cumbersome when developing web pages for different browsers. In this project this was never a problem due to the choice of only developing for the open source code browser, Mozilla Firefox. Another disadvantage is the steep learning curve when one has to master a combination of several technologies. To help developers many sites offer ready made DHTML solutions, but one should be aware of the compatibility problems that can arise if not used properly.[38]

6.2 The Server and the server-side techniques

Operations that are performed by the server are referred to as server-side operations. In this master thesis Tomcat was chosen as the web server to interact with one or more clients. This was due to the fact that Tomcat has become the standard for JavaServer Pages, and also the one Sun uses when creating its JSP implementation. Operations on the server-side also include processing and storage of data all of which are reachable to the client.

6.2.1 Tomcat

Tomcat is a web server that was developed by Sun and later donated to the Apache Software Foundation. Tomcat is increasingly used as a standalone web server in high-traffic and high-availability environments. It is written in Java so it runs on any operating system that has a Java Virtual Machine and it compiles JSPs into Servlets.

Due to the open source code, members of the Apache Software Foundation and independent volunteers develop and maintain Tomcat. It is an open development project that intends to assemble the expertise from developers all over the world.[39]

There is nothing in Tomcat's design or implementation that prevents it from being used in a validated environment. For example is Merck (Merck & Co. is a global pharmaceutical company that discovers, develops, manufactures and markets vaccines and medicines.[40]) using Tomcat in a validated environment. [41]

Security wise, it would be beneficial to gather all access points into one access point, i.e. the web server. Using this solution it is much easier to analyse the loopholes and address them appropriately. Tomcat's security record is also impeccable. There have been no public cases of damage done to a company, organization, or individual due to a Tomcat security issue. There have been no documented cases of data loss or application crashes caused by an intruder. [42]

It is also possible to use OpenSSL and to set up an own CA and server certificates. [43]

Directory structure
The typical and default directory hierarchy of a Tomcat installation comprises the following:[44]

- bin – startup, shutdown and other scripts and executables
- common – common classes that Catalina and web applications can use
- conf – XML files and related DTDs to configure Tomcat
- logs –application logs
- server – classes used only by parts of Tomcat
- shared – classes shared by all web applications
- webapps – directory containing the web applications
- work – temporary storage for files and directories

In this master thesis version 5.5 was used.

6.2.2 What is a Servlet?

A Servlet is a Java programming language class that allows software developers to add dynamic content to a web server. This extends the capabilities of servers to communicate via the request-response model, namely the HTTP protocol. In our case a web server hosts applications for dynamic use and for these the Java Servlet technology defines HTTP-specific Servlet classes, `javax.servlet` and the `javax.servlet.http` package. The `javax.servlet` package defines the interaction between the web container and the Servlet while the `javax.servlet.http` package defines HTTP subclasses of the generic Servlet.[45] Figure 6.1 shows a schematic model representing the communication procedure depending on if the Servlet has already been loaded or not.[46]

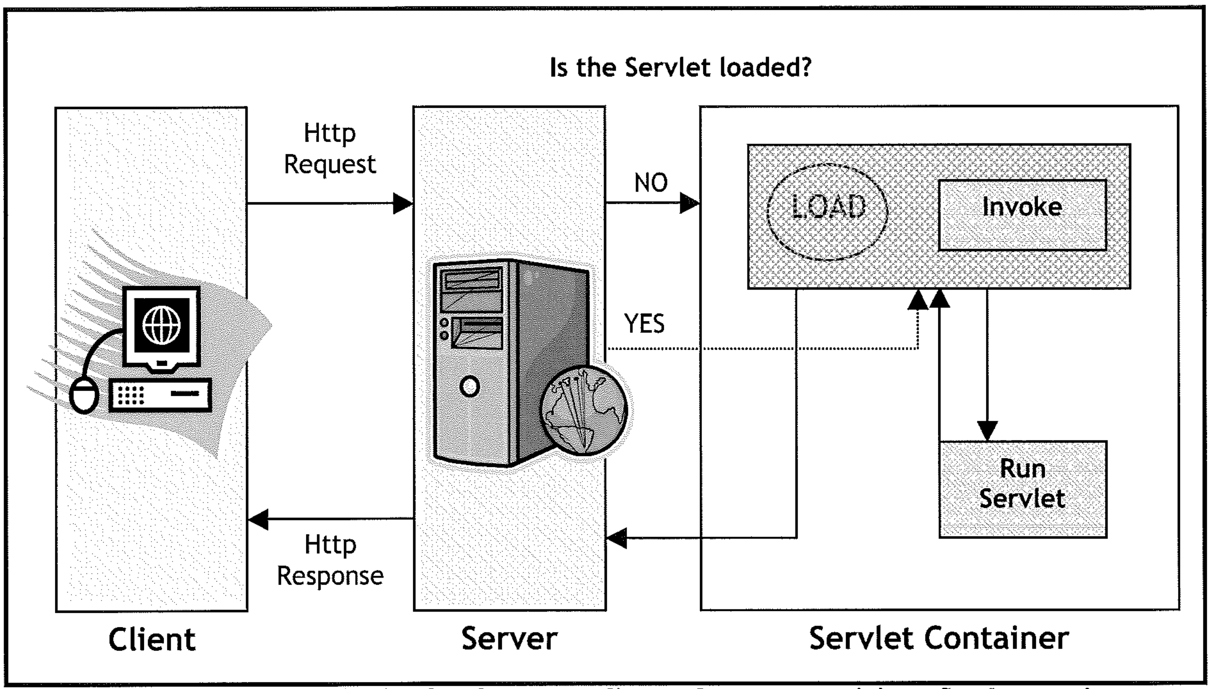


Figure 6.1 The communication flow between a client and server, containing a Servlet container.

Specific for Servlets are:

- They have to implement the Servlet interface, which defines life-cycle methods.
- The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.
- The HttpServletRequest, HttpServletResponse and the HttpSession define the communication between the web server and the client.

The generated content is often HTML, but can be other data as well. Servlets can maintain state across many server transactions by using HTTP cookies, session variables or URL rewriting.[47]

Security

A Servlet uses status codes and HTTP headers to manage its own security policy. Authentication and authorization is provided through a declarative security framework. This means that restricted resources and authorized users are not hard-coded into the application. Instead, a configuration document specifies the types of users to whom the resources are available. Thus, security policies can be changed easily according to requirements.[48]

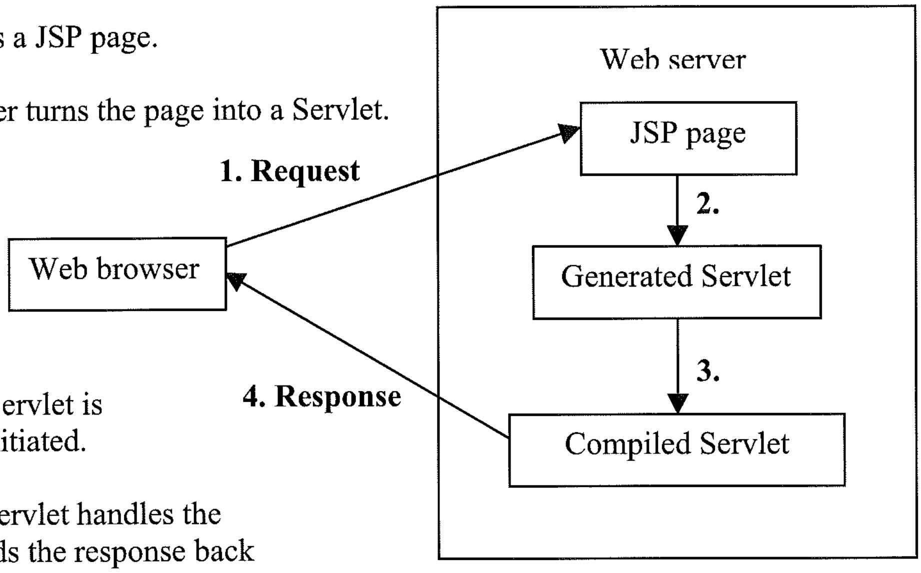
6.2.3 JSP

JSP is an abbreviation for JavaServer Pages. It is a simple but still powerful technology that allows software developers to generate dynamic HTML from a web server. One of its greatest benefits is the capability of separating the front-end presentation from the logic [49], which means separating the coding from the graphical design. To invoke built-in functionality JSP is equipped with certain syntax, called actions (see Appendix A for more information about JSP actions). To extend the capabilities even more JSP supports the creation of JSP tag libraries, which are extensions to the standard HTML and XML tags.

JSPs are compiled into Java Servlets by the JSP compiler. The source code to the *.jsp file is stored in a catalogue that is a part of the web server's catalogue structure. When a client requests the file, the web server will recognize the suffix and treat it in a special way. Listing 6.1 shows the different phases. The first time a client asks for it, it will be compiled into a Servlet and stored in RAM (Random Access Memory). The result will then form the response that is sent back to the client. If the client requests the same page several times and it has not been changed, the web server will use the same already compiled object.

Listing 6.1 Phases during a request

1. A client requests a JSP page.
2. The JSP compiler turns the page into a Servlet.
3. The generated Servlet is compiled and initiated.
4. The compiled Servlet handles the request and sends the response back to the client.



Servlets vs. JSP

Both Servlets and JSP provide the possibility to present dynamic web content and they have complementary API's. One of the major drawbacks with Servlets, is that it is not object-oriented. JSP is abstracted one level and can therefore contain Java program fragments which instantiate and execute Java classes. Often these two are used in combination with each other to form a unified compatible framework.

More reasons for choosing JSP can be the difficulties one can experience with maintaining the HTML code within Servlets. Other benefits why one should choose JSP are: plattform independency, easier to author, recompile automatically when changes are made to the source file, supports software reuse through the use of components, simplify development with JSP, JavaBeans and custom tags.

6.2.4 JavaBeans and JSP

When creating dynamic web pages a combination of JavaBeans components and JSP technology is preferable. This combination makes it possible to store and pass data between the different web pages which in turn lead to a more dynamic site design.

The JavaBean needs to have accessor and mutator methods to handle the data, where `get()` is the accessor and `set()` the mutator method. Accessor comes from accessing data while mutator mutates, or changes, data.[50]

6.2.4.1 What is a JavaBean?

A JavaBean is a Java component which can operate on all Java Virtual Machines. They are defined as reusable software components that can be manipulated visually in a builder tool.[51]

This means that JavaBeans enable developers to write reusable components that can run everywhere. Other benefits are therefore their portability and platform-independency.

To be able to use, reuse, replace and connect JavaBeans there are certain conventions that must be followed. These are:[52]

- JavaBeans have to support JDK 1.1 or later serializable models. Serializable means that it will be able to persistently save and restore its state.
- It should have a no-argument constructor.
- Its properties should be accessed using get, set and other methods following a standard naming convention.
- It should contain any required event handling methods.

The above mentioned together forms the standard for JavaBeans. Furthermore, a JavaBean can be loaded into any program that supports JavaBeans and permits inspection and changing properties.

6.2.5 Error handling in JSP

Errors do occur during a JSP page's life cycle. It can occur during the first time it is loaded, translation time error, or during a request later on, request time error.

- **Translation time error**
  The first time a page is loaded the JSP source code transforms into the corresponding Servlet class. Errors can occur and are in that case often a result from problems during compilation. They are reported to the client as a JSP error 500. These kinds of errors are taken care of by the JSP engine.

- **Request time error**
  This kind of error occurs during run time and within the JSP page instructions or in components which have been requested from the JSP page. An exception is thrown but can be eliminated if caught. If not, the client's request together with the error will be forwarded to an error page.

In this master thesis simple error handling was used. The content of errorPage.jsp can be viewed in Listing 6.2. To redirect to the error page in case of an error one has to insert the following code snippet in the working .jsp page.

```
<%@page errorPage="errorPage.jsp" %>
```

Listing 6.2 errorPage.jsp

```
<html>
<body text="red">
<%@ page isErrorPage="true" %>
Error: <%=exception.getMessage() %> has been reported.
</body>
```

```
</html>
```

6.3 Other techniques

To be able to establish a communication facility between the PushServlet and the client a framework named Ajax was used.

6.3.1 AJAX

Ajax stands for Asynchronous JavaScript and XML and is a way for creating interactive web applications. It is not a technique but rather a new way of combining existing techniques for a new purpose.[53] According to WEB 2.0 visions Ajax supports its aims of developing more interactive and more user friendly web pages. Using Ajax users experience web pages more responsive since small amounts of data can be exchanged with the server resulting in different parts of a page being reloaded separately.[54] This is the major advantage of Ajax where the whole page is not reloaded, saving both time and bandwidth. Ajax appears like an extra layer between the web browser and the web server. From there Ajax can communicate with the server and manage data in the background. The result is shown directly on the page without the need of refreshing the whole page.

Another advantage of Ajax is the asynchronous part. For example, a server request can be done even before the user claims the data. Retrieving data beforehand increases both the usability and speed. The parts of Ajax that is used to send server requests and handle data are written with JavaScript while XML is used to code and structure the information sent between server and client.

In this project Ajax enabled the communication between the client and the PushServlet. This includes the following actions: Register to the PushServlet, subscribe and unsubscribe to different topics, publish new data and listen after new and updated data. More on how these methods work and how they are connected can be found in Chapter 9.1.

7 System Communication

The way the system communicated in the past was a mixture of different techniques including several databases, see Figure 7.1. The information stored in the database was; process data (P data), process visualisation data (PVI), configuration and different recipes for how the machine is going to function. The HMI Screen was communicating both with the Process data database, configuration database and the PVI Communication Library. One major disadvantage with this solution is the maintenance. It is hard to maintain because of the complex architecture with many different user applications.

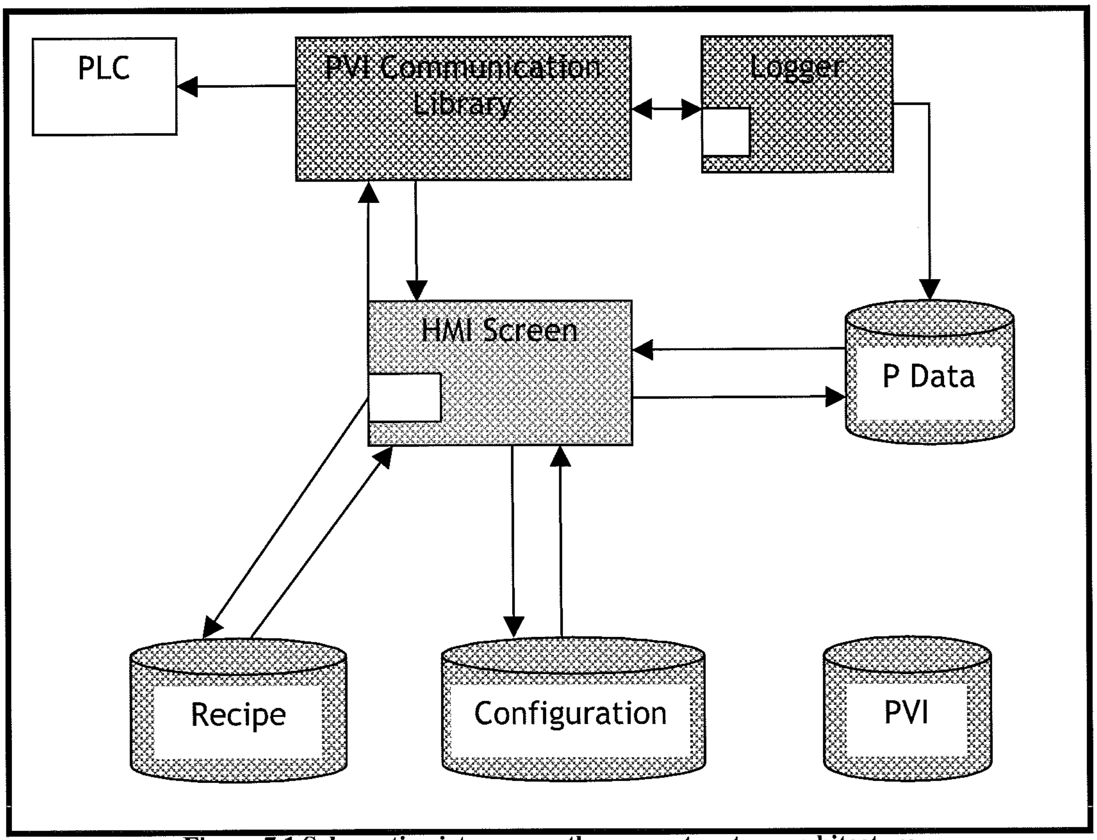


Figure 7.1 Schematic picture over the current system architecture

The new solution is less complex than the current one, see Figure 7.2. The HMI Client is communicating with the PLC through the Pusher, which also is part of the HMI Engine.

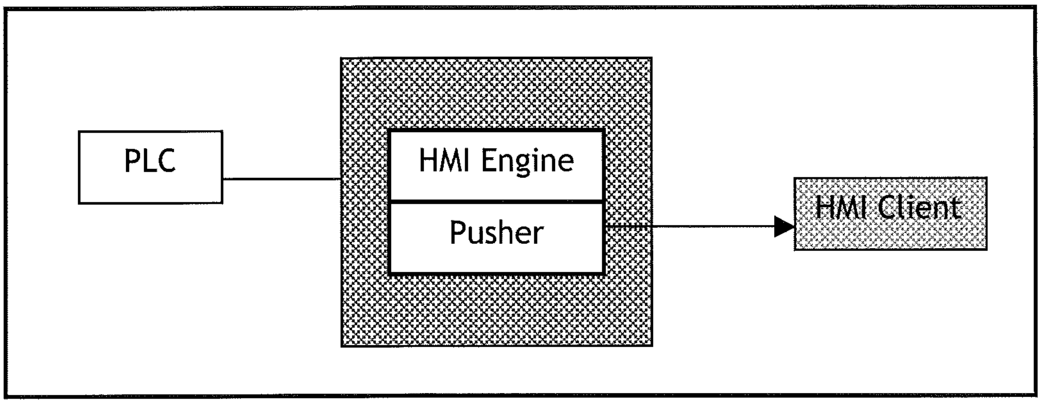


Figure 7.2 Schematic picture over the new system architecture

The next step in the process to simplify the system architecture is to remove the engine. The need of the engine is unnecessary, since the Java programs can communicate directly through RMI, this according to Maja Arvehammar's master thesis *Object-oriented automation system.* In the thesis she also claims that there are plenty to gain in the area of communication when introducing object oriented languages. Languages like Java, C++ and C# include extensive standard libraries, inheritance possibilities and ready-made design pattern solutions for common problems. There are also other advantages for example being able to structure and specify the software solution modelling.

7.1 Server and Client Communication

There are different ways for a client to communicate with a server. RMI makes it possible for clients to access objects that are located on a server. This brings about many advantages for a programmer, who can refer to methods and objects as if they were local.

HTTP is important for communication over the web and it defines how the communication between servers and clients should work. There is however a problem with HTTP, because of its nature of being a stateless protocol. This means that it forgets that a conversation between a client and a server ever took place. To solve this problem HttpSession object can be used. In this master thesis, HttpSession objects were used when implementing the Master solution design, see Chapter 9.3.

7.1.1 RMI

RMI stands for Remote Method Invocation and is one of the simplest ways of distributed computing with Java. The technique makes it possible for clients to execute remote method calls on a server using a transparent interface. In other words, RMI provides a way for interactions between objects working on different virtual machines on other networked computers. If one looks at it from a client's perspective, it can reference objects on the server as if they were local.

In a RMI system the connection between a client and a server is layered. In Figure 7.3 one can see the RMI system and how the connection becomes transparent because of the RMI subsystem.

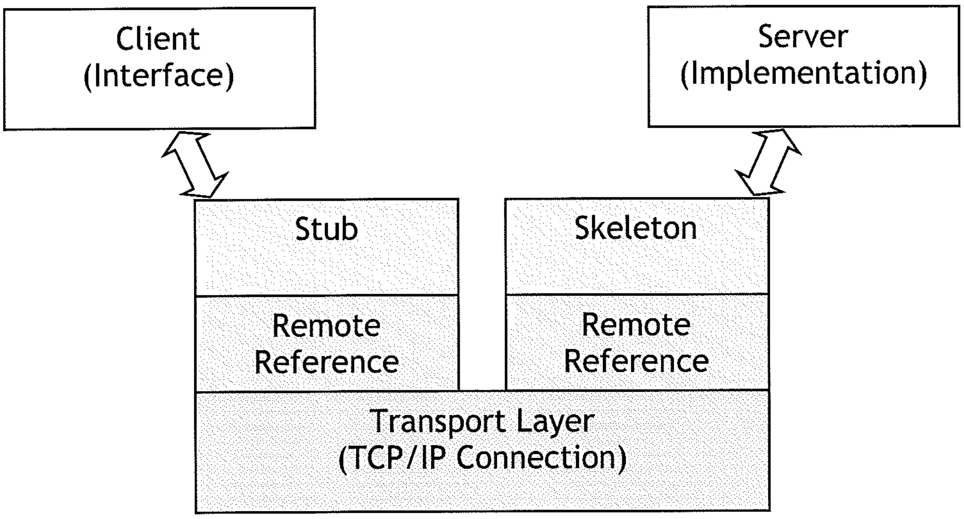


Figure 7.3 The RMI connection system. The client uses the stub and requests a reference to an object from the server side, while the server receives the request on the server side, via the skeleton.

The advantages of using an RMI approach are many. One is the level of abstraction the stubs and skeletons can provide, giving the developer the opportunity to write code as if all methods and objects are local to the client. [55]

How does it work?

When a client tries to execute a method provided by the server it uses the stub, which executes the actual implementing class of that service. The only requirement is that objects must implement the java.io.Serializable interface and all methods to a remote method must be either Java primitives or implement that interface.

The remote reference layer (RRL) handles reference variables to remote objects, using the transport layer's TCP/IP connection to the server. In other words the RRL negotiates the requests by converting objects into portable form across the network, marshalling. If the data does not fulfil the stated requirements for the communication, the stub will throw a MarshallException.

7.1.2 HTTP

Hypertext Transfer Protocol, HTTP, is one of the most important protocols that are being used for communication over the web. HTTP defines a set of rules how messages and other data should be written and how the communication between servers and clients should work. Communication is set up when an HTTP client establishes a TCP connection to a particular port on a remote host. On the other side an HTTP server listens on the specified port and waits for a request from the client. When a message is received a status line and a message is sent back to the client. A sample conversation between an HTTP client and an HTTP server running on www.example.com, port 80 can be seen in Listing 7.1.[56]

Listing 7.1 HTTP Sample

Client request (followed by a blank line, so that the request ends with a double newline, each in the form of a carriage return followed by a line feed):

```
GET /index.HTML HTTP/1.1
Host: www.example.com
```

Server response (followed by a blank line and text of the requested page):

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/HTML; charset=UTF-8
```

HTTP is also the protocol Ajax uses for sending requests. Ajax components have to be able to create HTTP requests to send to the server whose answer will decide the next action. A HTTP server should at least implement the GET and HEAD methods and if possible also the OPTIONS method.[57] Appendix D explains the different HTTP request methods in more detail.

7.1.3 HTTP Security

The `GET` and `HEAD` methods are defined as safe, which means that their usage should not result in any side effects, like changing the state of the server. `POST`, `PUT` and `DELETE` are stated as unsafe methods and should therefore be presented to the user in a special way.
Even if the GET methods declare to be safe there is a risk for it to cause changes on the server.[58]

7.1.3.1 HTTPS

HTTPS is used for ensuring a secure communication on the web. It is a URI scheme which is used when accessing resources with an extra encryption/authentication layer between HTTP and TCP.

The right implementation of the Secure Socket Layer, SSL, gives reasonable protection against attacks like eavesdropping or man-in-the-middle attempts. In other words, SSL supplies HTTP communication with an encrypted transport mechanism.[59]

Implementation
A secure communication is only possible if all units concerned are prepared to accept HTTPS connections. A public key certificate must be created for the web server, which in turn must be signed by a certificate authority. On Tetra Pak's behalf there is no certification done within the intranet, but introducing this solution would result in a proposal using OpenSSL for encryption. This is easily implemented for increasing the system protection.

One can not assume that by using only HTTPS one is fully protected. The system is not more secured than the weakest device, which means that other security actions need to be implemented. Due to a well-developed security implementation on behalf of Tetra Pak there is a high level of protection.

7.1.4 HTTP Session Object

The `HttpSession` object makes it possible to share objects across other JSP pages. As mentioned before HTTP is based on a request and response protocol. As soon as the application sends the response back to the browser, it looses all data concerning that conversation. There is however a solution to this problem, but it needs special processing if any information is to be kept while the user navigates through the site.

The solution is the `HttpSession` interface, which provides a way to store and manipulate session data that is available to all Servlets of the web application while the session remains valid. As long as the session is valid a reference to the objects inside of the session exists. The session can die, either by become invalidated, expire due to a time-out or if the application goes down (crashes or is undeployed). When the session is no longer valid due to for example a session timeout, the objects within are flagged for garbage collection.

To be able to tell different session objects apart, session IDs are used. They are stored in the browser as a cookie, but can also be maintained by URL rewriting if cookies are unsupported by the browser. On a client’s first request, the Container generates this unique session ID and gives it back to the client with the response, see Figure 7.4. The client sends back the session ID with each subsequent request. The Container sees the ID, finds the matching session, and associates the session with the request, see Figure 7.5.[60]

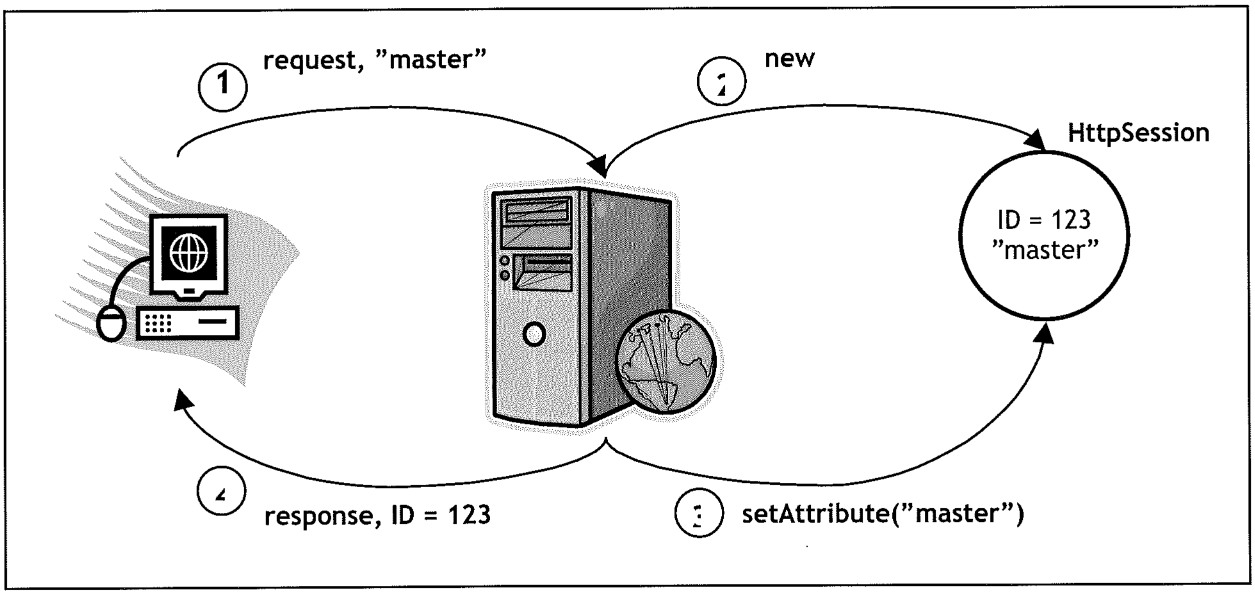


Figure 7.4 First request session

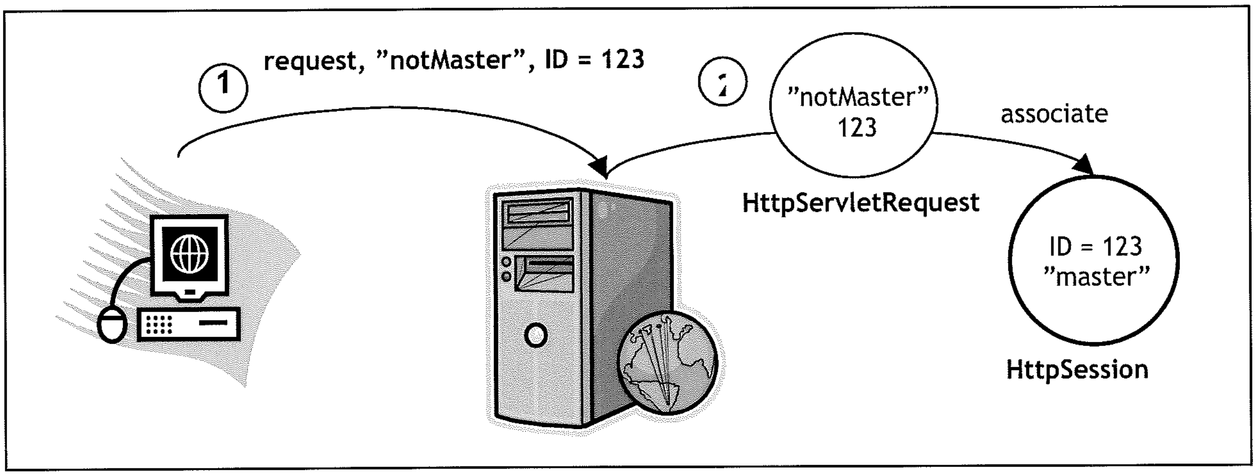


Figure 7.5 Subsequent request session

More explicitly, the programmer informs the Container to create or use a session. The rest will the Container run; generating the session ID, creating a new Cookie object and put the ID into it, and finally set the cookie as a part of the response. This procedure concerns a new session. If it is a subsequent request the Container gets the session ID from a cookie, matches the session ID with an existing session, and associates that session with the current request. The programmer does never see the session ID, but it can be retrieved if needed.

In this master thesis HttpSession objects were used when implementing the master solution. In Chapter 10 this is described in more detail.

There are several reasons why one should store state in the web tier.[61]

- **Easy implementation** - Since the application server handles the implementation of HttpSession, the developer is freed from handling with the details of designing, implementing, and testing code for managing session state.

- **High-quality, efficient operation** - An application server's HttpSession implementation is optimized and tested for that server, and therefore will probably be more efficient and reliable than a custom solution.

- **Potentially richer feature set** - An application server's implementation of session state management may include such features as failover, cluster support, and so on, that go beyond the base-level requirements of the J2EE platform specifications. The system architect can select a server platform with the differentiating features that best suit the application requirements, while maintaining J2EE technology compatibility and portability.

- **Portability** - The HttpSession interface is standardized, and so must pass the J2EE Compatibility Test Suite (CTS) across all J2EE-branded application servers. Read about the role of the CTS and J2EE branding to ensure portability at http://java.sun.com/j2ee/compatibility.html.

- **Scalability** - When the application is allowed to manage session state by way of HttpSession, it can most effectively manage storage of that state in caches and/or server clusters.

- **Evolvability** - Application server vendors are constantly improving their offerings. Servers will maintain existing interfaces for backward compatibility, even as they add features that improve performance and reliability. An HttpSession implementation that works properly today will work better tomorrow as improved server versions become available, with little or no change to the source code.

The drawbacks with this approach are not many, but one can mention the limitations to the web client. A reimplementation of session state management will become necessary if clients require other types of communication than HTTP. Most servers persist session state or provide failover in case of a web container failure (i.e. restart or crash), but not all. This can result in an invalidation of all sessions in progress, losing all associated states.

8 Separating front and back end programming

In this section a further explanation about the techniques, that can be used when separating a programmer's work from a web designer's, will be described. First and foremost custom tag libraries will be discussed. This leads to a discussion about the benefits of a better look and feel with Cascading Style Sheets, and what Tetra Pak would gain from it. Finally the advantageous usage of design patterns and how it can improve system architectures is examined.

8.1 Custom Tag Libraries

The JSP platform includes a rich architecture for developing tag libraries. The technology has made it possible to implement anything that can be described by HTML or XML tags. Web page designers do not have to be programmers and with the introduction of custom tag libraries this can become reality.

Custom tags are both more advanced and also more flexible than typical JSP tags. One of the advantages is that they can communicate together with other tags on the web pages. Together a few simple tags can build a more complex process collectively. It is also possible to use the output of one custom tag as the input for another. It is desirable for the sake of usability to break down the processes to smaller components.

The custom tag library is a powerful feature of JSP v1.1 that aids the separation between logic and presentation. JSP v1.1 technology supports the development of reusable modules called *custom actions*. A custom action is invoked by using a *custom tag* in a JSP page. A *tag library* is a collection of *custom tags*.[62]

Some features of custom tags are:

- They can be customized via attributes passed from the calling page.
- They have access to all the objects available to JSP pages.
- They can modify the response generated by the calling page.
- They can communicate with each other. One can create and initialize a JavaBeans component, create a variable that refers to that bean in one tag, and then use the bean in another tag.
- They can be nested within one another, allowing for complex interactions within a JSP page.

There are two types of components for a tag library: the tag library descriptor file and the tag handlers. With these a JSP is able to use tags contained in the library within its page[63].

The TLD File

A tag library descriptor (TLD) file is an XML document that describes the library. A TLD contains information about the library as a whole and about each tag contained in the library. TLDs are used by a JSP container to validate the tags. Parts from `TetraTagLib.tld` can be found in Listing 8.1.

Listing 8.1 An extraction from TetraTagLib.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library
1.1//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>

    <tlibversion>1.0</tlibversion>
    <jspversion>2.0</jspversion>
    <shortname>tetraTagLib</shortname>
    <info>Tetra Tag library</info>

    <!--  The push tag. -->
    <tag>
        <name>push</name>
        <tagclass>com.tetrapak.rd.dev.webHMI.DoPushTag</tagclass>
        <bodycontent>empty</bodycontent>
        <attribute>
          <name>topic</name>
          <required>false</required>
        </attribute>
    </tag>

    <!-- This tag adds the alarm display -->
    <tag>
        <name>addAlarmDisplay</name>
        <tagclass>com.tetrapak.rd.dev.webHMI.AddAlarmDisplayTag</tagclass>
        <bodycontent>empty</bodycontent>
    </tag>

    ....
    ....
    ....

</taglib>
```

The Tag Handler

The tag is defined in a handler class. `TagSupport` is the base class used for simple tags. It can be found in the `javax.servlet.tagext` package. What your tag is implementing will depend on what methods could potentially be called and what needs to be implemented. In Listing 8.2 one of the tag handler classes can be found, `DoPushTag.java`.

Listing 8.2 DoPushTag.java

```
package com.tetrapak.rd.dev.webHMI;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class DoPushTag extends TagSupport{

      private String topic = null;

      public void setTopic(String value){
           topic = value;
      }

      /**
      * doStartTag is called by the JSP container when the tag is
      * encountered
      */
      public int doStartTag() throws JspException {
           JspWriter writer = pageContext.getOut();
           StringBuffer sb = new StringBuffer();

           try {
               sb.append("<div id='push' value='"+topic+"'>");
               writer.print(sb.toString());
           } catch (Exception ex) {
               throw new JspTagException(ex.getMessage());
           }
           return SKIP_BODY;
      }

      /**
      * doEndTag is called by the JSP container when the tag is encountered
      */
      public int doEndTag() {
           try {
               JspWriter out = pageContext.getOut();
               out.println("</div>");
           } catch (Exception ex){
               throw new Error("Could not evaluate end tag.");
           }
           return EVAL_PAGE;
      }
}
```

8.1.1 Designing a custom tag library

When creating a tag library producers have to think about the final consumers. Dividing the work between different actors can be a complicated task when the both sides do not speak the same language. The complexity of the problems increases when it is the programmer's task to design a language which has to be understood by a web page designer. Programmers are versed in a procedural programming paradigm while web page designers operate differently.

To succeed, the programmer needs to be consistent and ask clients if they understand the semantics. The usage of self explaining code facilitates this work, but there are still communication barriers to overcome.

8.2 Design Patterns

Design patterns are class- and method-level solutions to common problems in object-oriented design. They are results of solving similar problems with common and effective methods in different contexts. Reusability is an important issue in today's software development, and with the usage of design patterns this can enhance to a higher degree. In this master thesis different patterns were used for different aspects of software. The Model-View-Controller pattern was used during the work of separating the front and the back-end programming, while the Observer and Mediator patterns were used to distribute responsibility. All three fulfil an important purpose.[64]

8.2.1 Model-View-Controller

Model-View-Controller (MVC) is a classic design pattern often used by applications that need the ability to maintain multiple views of the same data. The advantages of using this design pattern are that the different subsystems can be developed and maintained separately. This means that it is possible to change the user interface without changing the model and the cohesion is higher since the logic and the user interface is separated, see Figure 8.1.

Unlike the other frameworks, J2EE defines a pattern for model objects.[65]

- **View** – The view in a J2EE application may be represented by a JavaServer Page. Alternately, the code to generate the view may be part of a Servlet.
- **Controller** – The controller in a J2EE application may be represented by a Servlet. The Servlet processes and responds to events and may invoke changes on the model and view.
- **Model** – The model is represented by entity beans. The model is another name for the domain layer. Domain logic adds meaning to raw data.

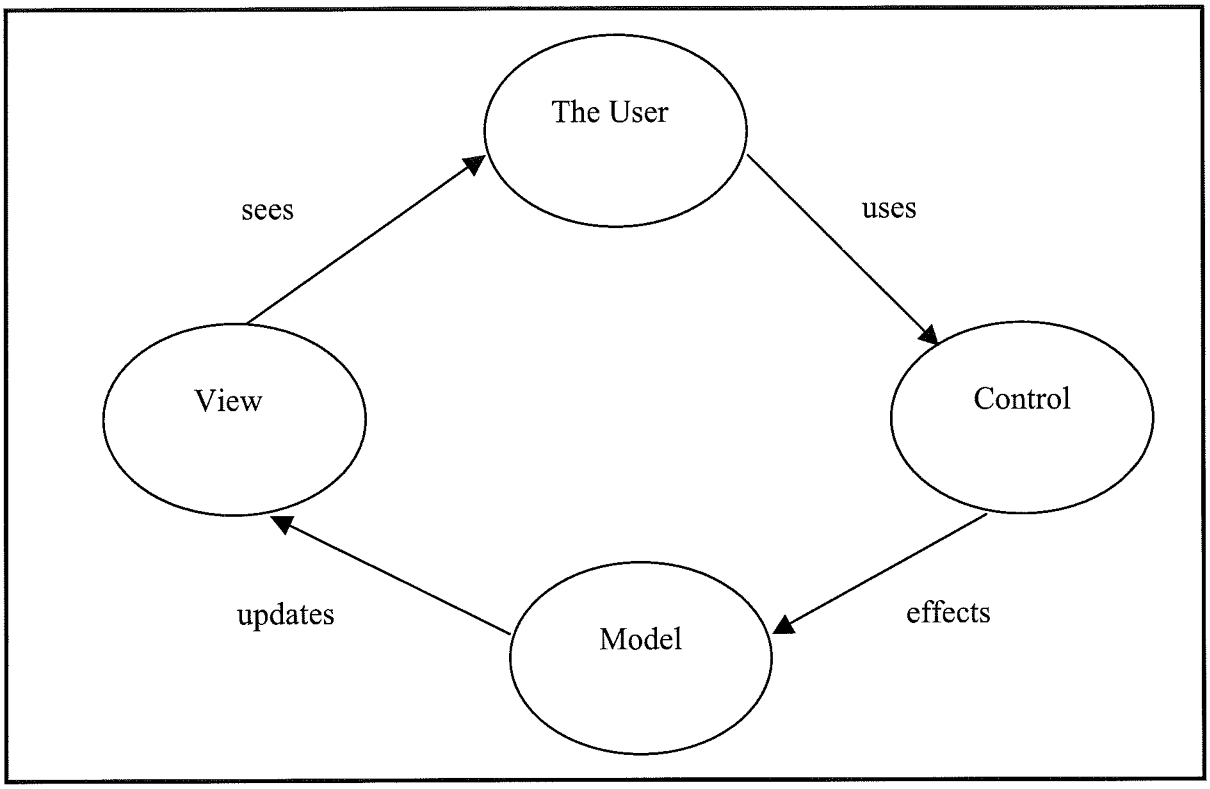


Figure 8.1 A model of the Model-View-Controller pattern

8.2.2 Observer

The Observer pattern is used for distributing responsibility. It minimizes the responsibility of an object that other objects find interesting. This pattern is sometimes known as publish/subscribe and is often used in computer programming to observe the state of an object in a program. These objects are registered to observe an event which may be raised by the observed object (the subject). A simplified picture of the Observer pattern's essence in this project can be seen below in Figure 8.2.

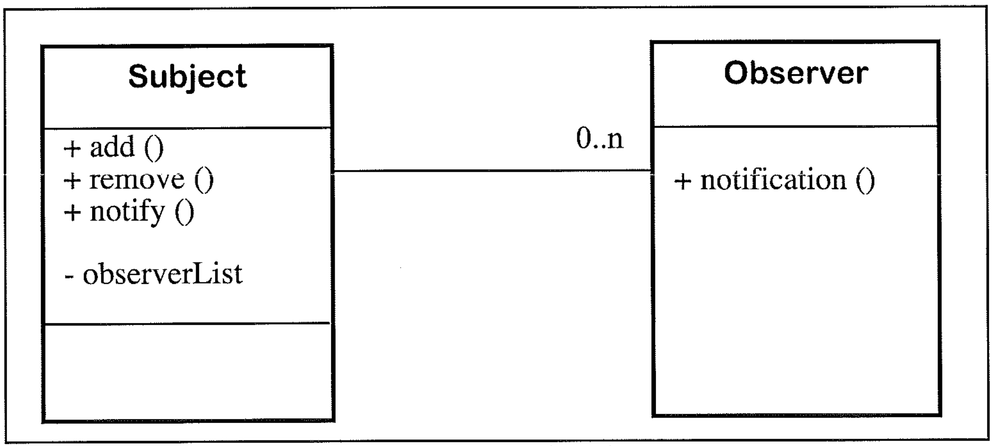


Figure 8.2 A simplified model of the Observer pattern.

When an event is raised each observer receives a call-back. This can be a virtual function of the observer class (see *notify()* in Figure 8.2) or a function pointer. Each observer implements *notify()* and at the same time defines its own behaviour when the notification occurs. The subject normally has a method for adding a new listener, *add()* and a *remove()* method for removing observers from the list.

One typical usage of the observer pattern is listening for changes of the value of a property of an object. Another characteristic is that it is very often associated with the MVC paradigm. Here the observer pattern is used to create a loose coupling between the model and the view. Typically, a modification in the model triggers the notification of model observers, which are actually the views.

8.2.3 Mediator

In contrary to the Observer pattern, which minimizes the responsibility, the Mediator pattern centralizes the responsibility. This is done for a particular set of objects rather than for all clients in a system. This pattern is commonly used when developing user interfaces with Java.

"The intent of the Mediator pattern is to define an object that encapsulates how a set of objects interact; this promotes loose coupling, keeping the object from referring to one another explicitly, and lets you vary their interaction independently."[66]

Mediator works like a central authority and makes a separation between the component-building, event-handling and mock database parts of the application possible. The structure of the mediator can be seen in Figure 8.3.[67] It consists of at least three objects messaging in a star topology.[68] In the Tetra Push solution the mediator is used in PushHandler. When receiving a value, the PushHandler transforms it into a script and forwards it to connecting colleague objects, in this case the HMI Engine and the PushServlet.

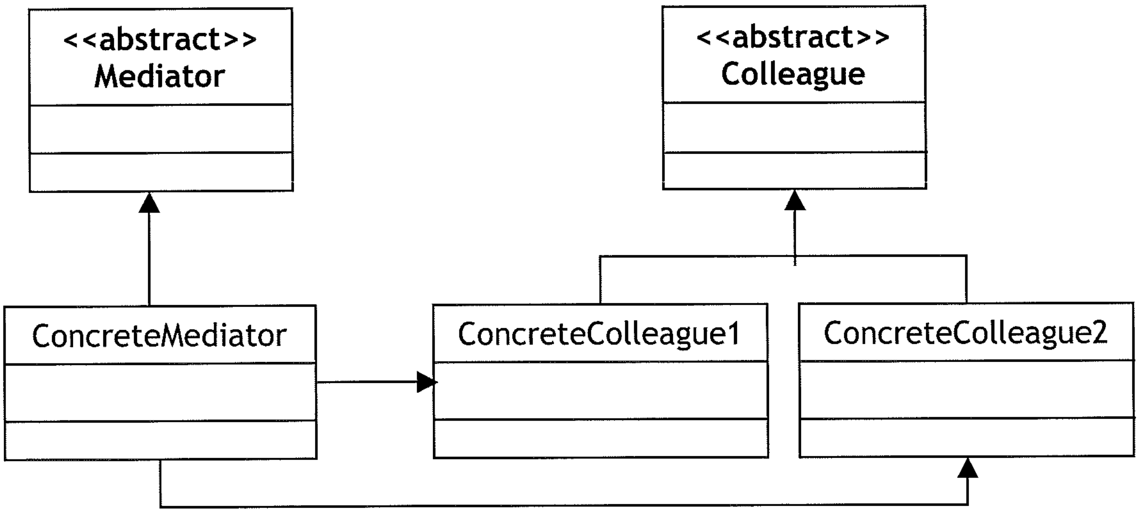


Figure 8.3 The structure of a generic mediator design class diagram. It shows the abstract mediator and colleague class.

As seen in the model there are only two types of classes in the pattern. The abstract Mediator class defines an interface for communicating with colleague classes and contains the basic methods. This means that the Mediator controls the interaction between the colleague classes, i.e. how the data transfer is conducted. The coordination of the colleague objects is done by the subclass ConcreteMediator, which implement the basic methods and adds more functionality to its interface beyond the basic. If an object wants to communicate with other objects this is done via its own Mediator, which will follow the set of guidelines for dealing with the incoming information.

Using the Mediator pattern makes it easy for a programmer to change the behaviour of the system, simply by replacing the concrete mediator class and/or some of the methods contained within it.[69]

Tetra Push uses the Mediator to control the flow of communication. One colleague object sends an output to the Mediator, while another requests an input. The Mediator's task is to organize this information and redirect the data flow between all colleagues.

8.2.4 Singleton

Singleton is probably the best-known pattern and is used to centralize authority in a single instance of a class. By using this pattern other objects relying on the responsible object can find it thanks to its property of providing a single point of access. One should have in mind that it is easily misused. This is due to the temptation of using it when creating global variables.[70]

In TetraPush Singleton was used to ensure a single instance of the server interface in the PushServlet.

9 Results

When developing a Web based HMI many aspects need to be taken into consideration. Our task was to make it possible to divide a programmer's work from a web designer's. This was partly done using JSP's rich architecture for developing custom tag libraries. In the next sections we will describe the Tetra Push system in detail which will be concluded with more specific results from the Web HMI. The general overview will have its code omitted, while our solution will be complemented with some code fragments to achieve a greater understanding of our work.

9.1 *Tetra Push*

Tetra Push is the name for the solution built upon a Servlet. In the following sections we will describe the Tetra Push implementation in more detail. We will also elucidate its pros and cons from a performance and security perspective.

The included parts of the TetraPush, except for the Web HMI, were developed simultaneous by our supervisor Mattias Wallinius if not stated otherwise.

9.1.1 System architecture

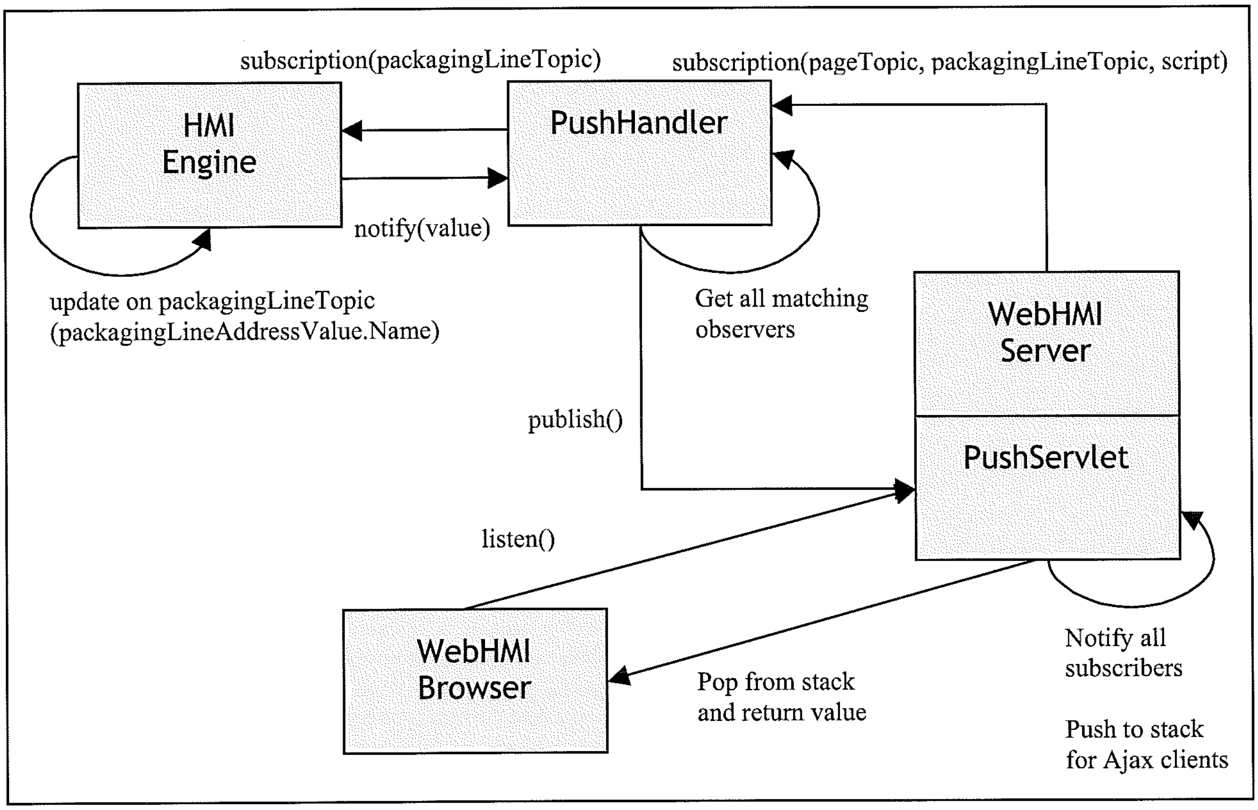

subscription(packagingLineTopic)
subscription(pageTopic, packagingLineTopic, script)
HMI Engine
PushHandler
notify(value)
update on packagingLineTopic (packagingLineAddressValue.Name)
Get all matching observers
WebHMI Server
PushServlet
publish()
listen()
WebHMI Browser
Pop from stack and return value
Notify all subscribers
Push to stack for Ajax clients

Figure 9.1 A schematic image over the Tetra Push system.

9.1.2 HMI Engine

The HMI Engine facilitates the communication between the PLC and the Pusher. It handles the PackagingLineAddressValues, adds the PushHandler as an observer to the values and notifies the PushHandler that a value has been changed.

9.1.3 PushHandler

The PushHandler was developed in parallel with this master thesis by Daniel Biloglav. To gain an understanding of its purpose and the right usage approach a couple of hours were spent with Daniel who showed us how to use it.

The PushHandler is responsible of handling all subscription functions to the HMI controller and notifications from the HMI controller to the HMI application. The subscription functions include both the 'subscribe' and 'unsubscribe' methods.

When a subscription is performed by a Web HMI it sends three parameters. The first parameter is called 'packaging line topic'. It stands for the name of the packaging line address value. The second is called 'page topic' which is the target of the notification message. The third one is the actual script, can be one or many, which will be executed on the target.

When the PushHandler receives a subscription from the Web HMI it adds the unique identifier of the active subscription in an internal list and directs a subscription request to the HMI controller. When a packaging line address value (which is included in the subscription) is updated in the HMI engine the HMI engine will perform a notification to the PushHandler attaching the updated packaging line address value. The PushHandler will then go through the subscription list and publish the new value with the matching JavaScript function and page topic of all concerned servers, see Figure 9.1.

9.1.4 PushServlet

The PushServlet uses Ajax in the communication with connecting clients.

There is a PushServer interface implemented in the PushServlet according to a Singleton pattern. This is due to the characteristics of the Singleton pattern where it ensures that there is only one instance of this class. Another feature is that it provides a global point of access which the PushServlet has to be aware of.

9.1.5 Tetra Push Actions

In this section the different actions in the Tetra Pusher system are explained briefly.

- **Subscription** – In order to gain notifications on event changes in the HMI engine a subscription must be performed. The request is performed on two levels. The first part of the request is defined between the Web HMI and the PushHandler. The Web HMI sends a request providing the PushHandler with the following information:
    - Page Topic – The destination for the push message, usually defined as an URL address.
    - Packaging Line Topic – The value the web HMI listens on. When this value is changed in the model, the HMI engine will notify the clients on the changed value.
    - Script – The actual HTML code or JavaScript that will be published on the web site.

The next part of the subscription use case is the request performed between the PushHandler and the HMI engine. With this request the PushHandler sends the full path of the PackagingLineAddressValue and is added as an observer to the value in the model.

- **Notify** – When a PackagingLineAddressValue is updated in the model, which is provided by the HMI engine, the updated value is sent to all clients.
- **Listen** – The Servlet pops data from the stack and returns it to the Ajax client who has sent an Ajax HTTP POST with the listen parameter. If the stack is empty NODATA is returned. The task of the stack is to buffer if the Ajax script has too low performance.
- **Publish** – Clients who listens to certain topics receives data when publish is called with the right topic parameter. This includes both push and Ajax clients.
- **Pop and return from stack** – see listen.
- **Push to stack for clients** – Ajax clients get an own stack in the PushServlet where data is saved until the clients asks for it.

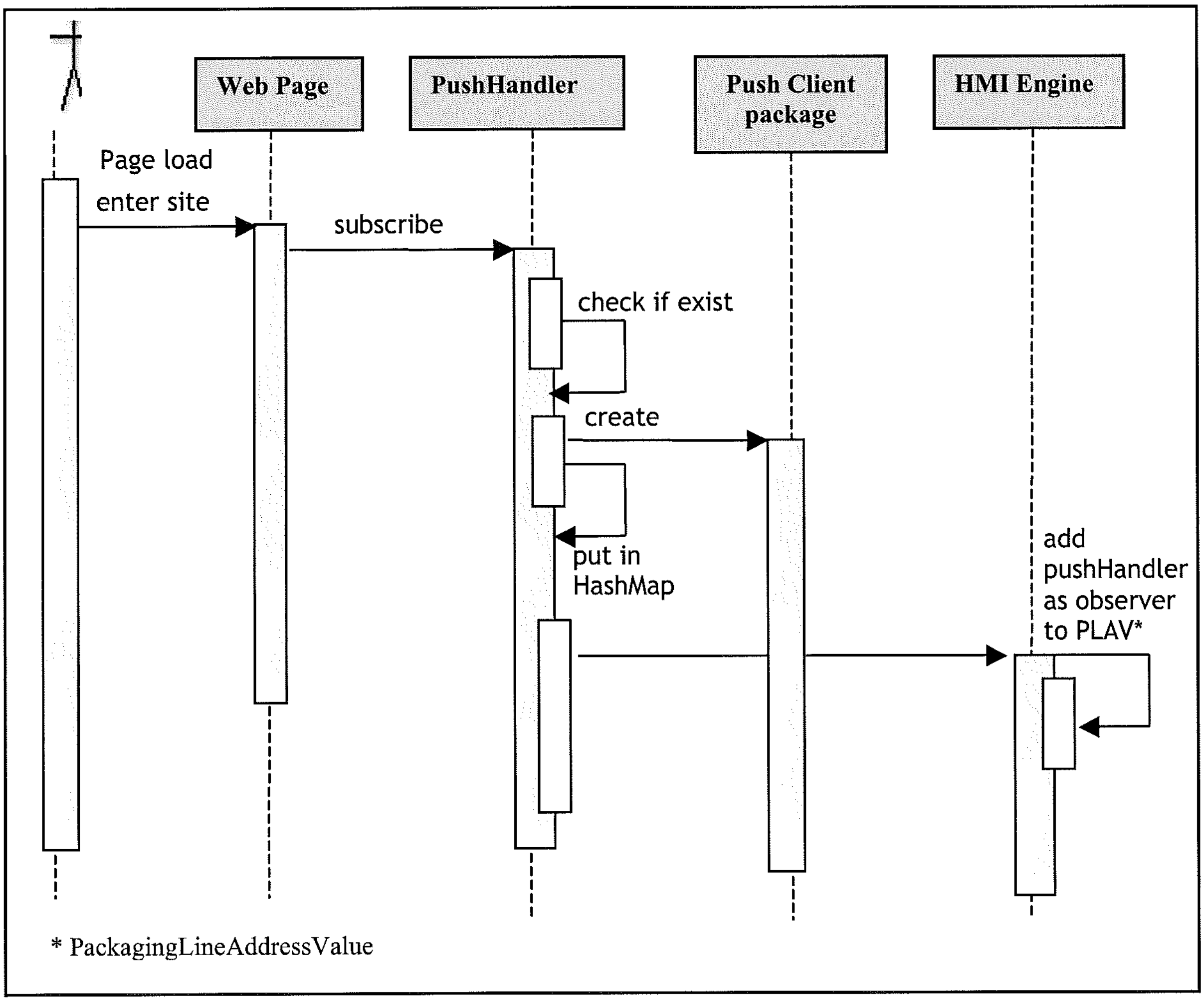
9.1.6 Tetra Push communication

The communication starts when a user enters the web site. The web page is loaded and sends a subscribe message with pageTopic, packagingLineTopic and script as parameters. When the PushHandler receives the request it checks if the client already exists, if not it creates a new Push client package and adds it to a HashMap. The HashMap contains a list of every client connected to the system that has made a subscription request. After that it forwards the subscription message to the HMI Engine but only sends the packagingLineTopic as parameter. This is the only thing the engine needs to know; which topics there are subscriptions on. Then the PushHandler is added as observer to the packagingLineAddressValue. By using the observer pattern, the PushHandler receives a notification when a value has been changed. When a notification appears, the value is broadcasted to the subscribers concerned. The broadcast contains the script which is published on the web page.

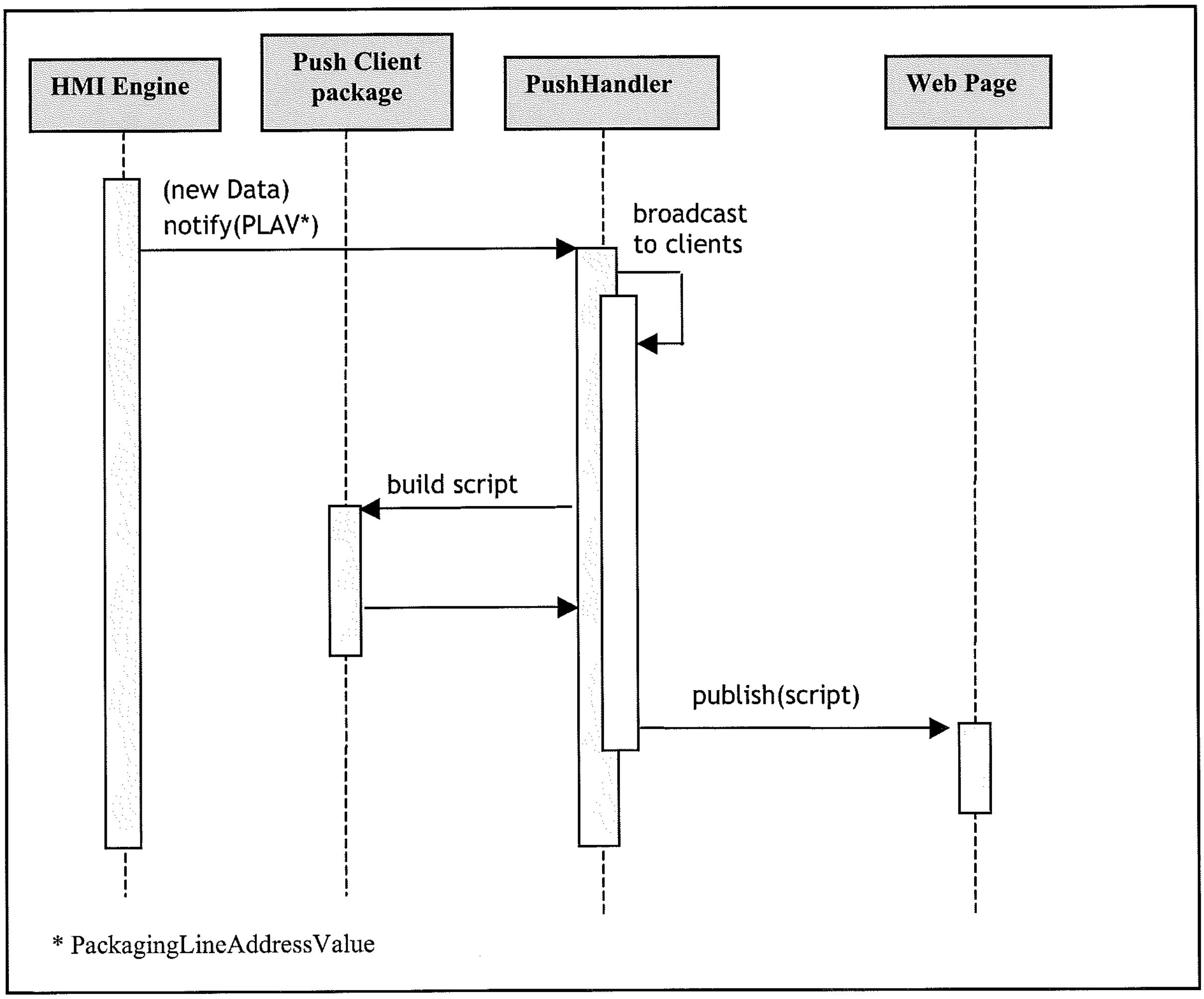
9.1.7 Web HMI Use cases

Below two different use cases are presented, showing a part of the communication of Tetra Push. The first one describes it on the basis of the user, and the second one from the HMI Engine perspective. The second one is a sequel to the first one, but is still presented separately due to the fact that the data flow is reversed.

Use Case 1: Subscribe



Use Case 2: Notify



9.1.8 Security of TetraPush

More specific security issues concerning the TetraPush are that it today does not use HTTPS. A first step would be to implement at least client authentication, which would be preferable and require only a small working effort. SSL is not supported by present Push clients. However, it is not possible to cut off the communication thanks to the TCP client socket which the client establishes with the server. Theoretically speaking there is a possibility to cut off the server's port, which is the result of a DNS attack. TetraPush will run inside the machines intranet and will not be exposed to this kind of attacks, and will also be protected in other ways accordingly.

9.1.9 Performance of TetraPush

There was a problem with a memory leakage that appeared after using the application for about one hour. The leakage was due to a problem with Firefox version 1.5.0.9 caching function that was not compatible with Ajax. The problem can be avoided by tuning the settings in Firefox.

Having the Pusher written in Eclipse it was easy to write test cases in JUnit. JUnit is an open source tool for project testing and debugging. [71] It is used for writing and run repeatable automated tests.[72]

Stress testing is vital for this kind of software. It is important that one knows the limit of the software and how it is supposed to be used. In this particular case both the maximum number of users and the size of the monitor were tested.

Apache JMeter was used to test the performance of this application. It was originally used only for testing Web Application but has now developed to test even other test functions. JMeter can be used to simulate a heavy load on the server, network or to analyse the overall performance under different load types. One of the advantages with using this program is the graphical representation of the result. [73]

In the context of Web applications, response time refers to the time elapsed between the submission of a request and the receipt of the resulting HTML.[74]

9.2 Web HMI

The basic idea for a Web HMI was developed in line with the Servlet solution. Due to time constraints the applet solution was not evaluated or tested together with our Web HMI, but the work done is presented in Appendix G.

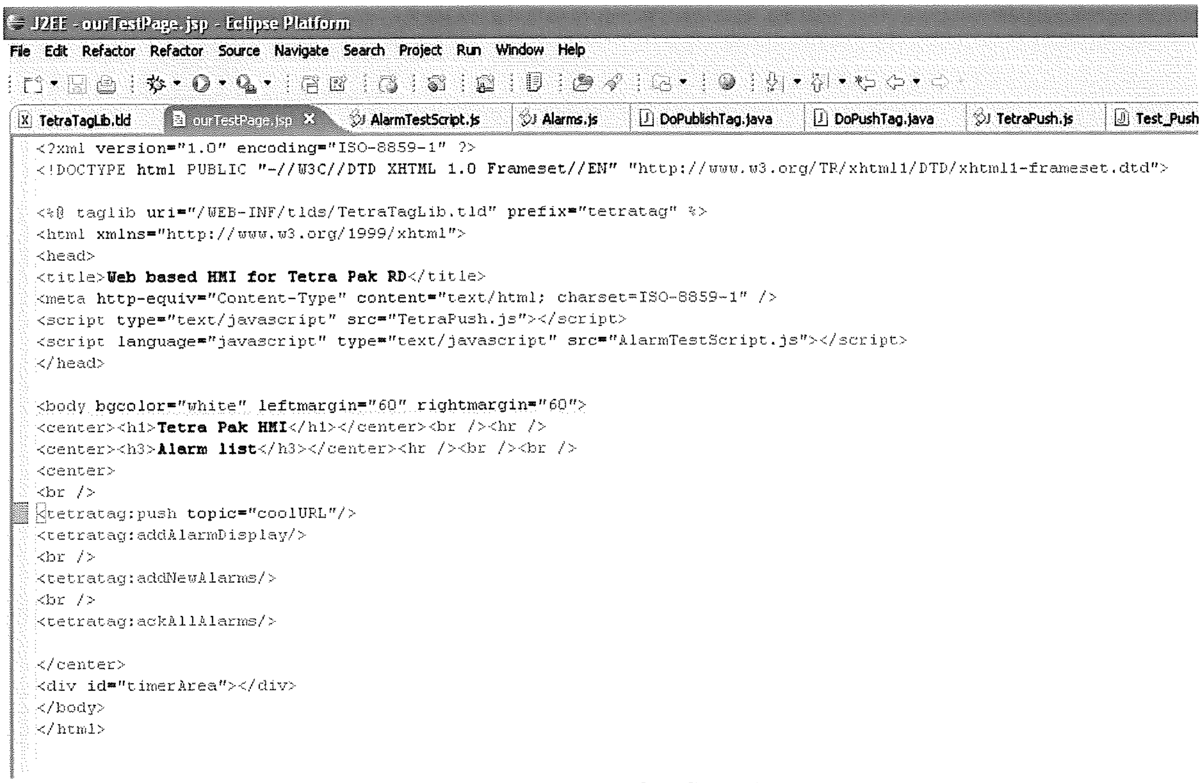
Both the Servlet and Applet solution function impeccable with the web HMI's usage of custom tag libraries, which had to be a matter of course. A description on how the Servlet solution used the functionality of custom tags will follow in 9.2.1.

9.2.1 Usage of Custom tag libraries

The separation between the work of a web designer and a programmer was possible due to the implementation and usage of Custom Tag Libraries. In the following sections we will take a closer look at the consequences this results in from the different actors' perspective.

9.2.1.1 Designer's perspective

When using a custom tag library a web designer needs to know very little about the technology behind the scenes. All one needs to know is how to take advantage of its strong capacities, in other words how to use it. With a well written description about the tags' name and their functionality a web designer has all the possibilities of taking the usage to a higher degree.
Figure 9.2 is an example of a designer's work space.



J2EE - ourTestPage.jsp - Eclipse Platform
File Edit Refactor Refactor Source Navigate Search Project Run Window Help
TetraTagLib.tld
ourTestPage.jsp
AlarmTestScript.js
Alarms.js
DoPublishTag.java
DoPushTag.java
TetraPush.js
Test_Push

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<%@ taglib uri="/WEB-INF/tlds/TetraTagLib.tld" prefix="tetratag" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Web based HMI for Tetra Pak RD</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<script type="text/javascript" src="TetraPush.js"></script>
<script language="javascript" type="text/javascript" src="AlarmTestScript.js"></script>
</head>

<body bgcolor="white" leftmargin="60" rightmargin="60">
<center><h1>Tetra Pak HMI</h1></center><br /><hr />
<center><h3>Alarm list</h3></center><hr /><br /><br />
<center>
<br />
<tetratag:push topic="coolURL"/>
<tetratag:addAlarmDisplay/>
<br />
<tetratag:addNewAlarms/>
<br />
<tetratag:ackAllAlarms/>

</center>
<div id="timerArea"></div>
</body>
</html>
```

Figur 9.2 A screen shot of a JSP code page .

The first line that differs from a normal HTML page is

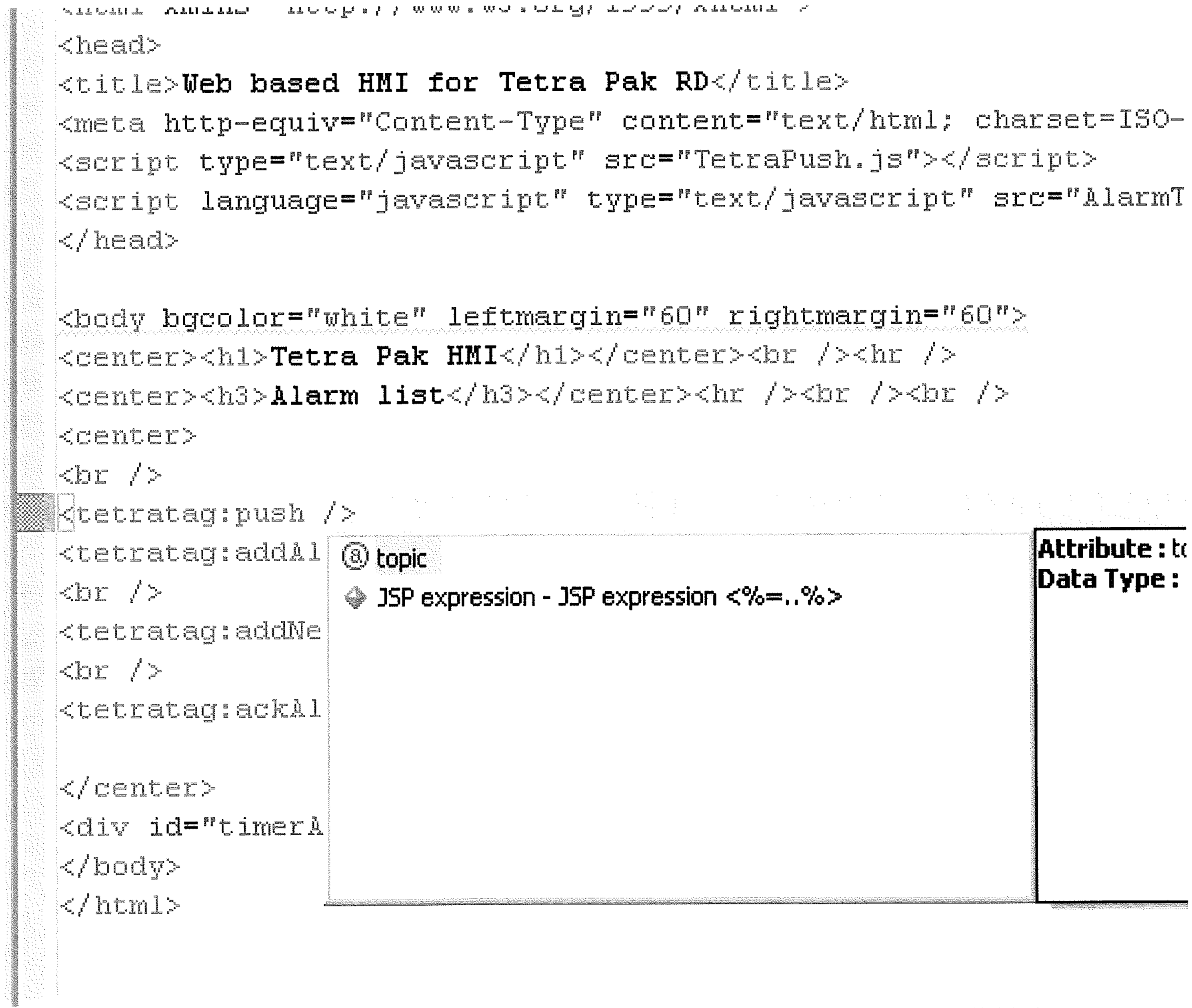
```
<%@ taglib uri="/WEB-INF/tlds/TetraTagLib.tld" prefix="tetratag" %>
```

If we break down the code above we can get an understanding about its contents and its functionality. The main purpose of this tag is to declare a custom tag library which we are going to use in our web page. This is done by declaring the library's name, TetraTagLib.tld written like a full URI, `uri="/WEB-INF/tlds/TetraTagLib.tld"`. A prefix is used to reference to the library without having to specify the whole URI every time. This information is wrapped between the JSP declare tag, `<%@ %>`.

When a correct declaration is done, one can start to use the tags defined in the tag library. The following code snippet shows how to use a custom tag:

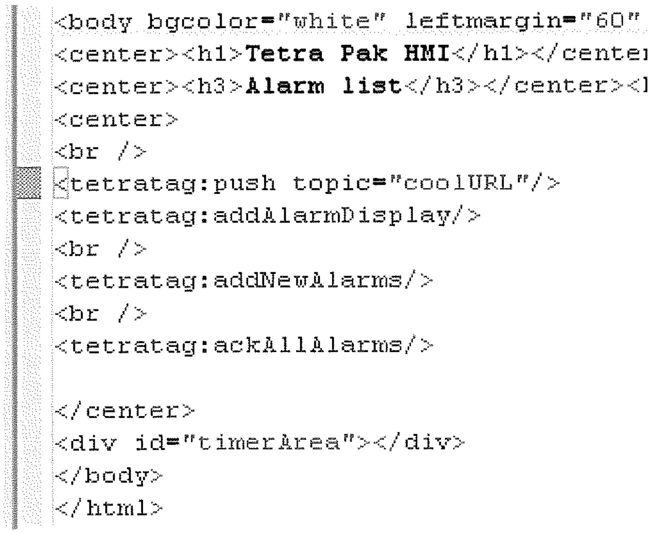
```
<tetratag:push/>
```

Some custom tags are declared with a body and/or with attributes. Depending on the structure the programmer takes different actions. In this project there were no tags with a body, however some with optional attributes. In Figure 9.3 an example of a `tetratag` with the `topic` attribute can be viewed, followed by the `tetratag` syntax in Figure 9.4. The code completion function helps the web designer to overview the possible attributes. Together with a well written documentation, code completion is enough to guide the web designer to the right and proper use of the custom tag library.



```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Web based HMI for Tetra Pak RD</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
<script type="text/javascript" src="TetraPush.js"></script>
<script language="javascript" type="text/javascript" src="AlarmT
</head>

<body bgcolor="white" leftmargin="60" rightmargin="60">
<center><h1>Tetra Pak HMI</h1></center><br /><hr />
<center><h3>Alarm list</h3></center><hr /><br /><br />
<center>
<br />
<tetratag:push />
<tetratag:addAl
<br />
<tetratag:addNe
<br />
<tetratag:ackAl

</center>
<div id="timerA
</body>
</html>
```

Figur 9.3 An example of code completion in Eclipse. Code completion is also available in Macromedia DreamWeaver.



```
<body bgcolor="white" leftmargin="60"
<center><h1>Tetra Pak HMI</h1></cente
<center><h3>Alarm list</h3></center><
<center>
<br />
<tetratag:push topic="coolURL"/>
<tetratag:addAlarmDisplay/>
<br />
<tetratag:addNewAlarms/>
<br />
<tetratag:ackAllAlarms/>

</center>
<div id="timerArea"></div>
</body>
</html>
```

Figur 9.4 A close up on tetratag syntax.

After including the desired tags the web designer's job is done. When the JSP page is viewed by the client it will present the logic the programmer has defined for the specific tag.

9.2.1.2 Programmer's perspective

The programmer's task is on the contrary from the web designer's more focused on the logic and technology which makes this approach possible. On one hand the programmer needs to be aware of how the technology is best used so the division of work, separating design and logic, becomes profitable. The programmer does not need to know the exact design of the page, but indeed what the purpose of the page is. Communication for achieving a mutual understanding of each areas main problem is an essential part for reaching the best result possible.

To facilitate the work for the web designer a programmer should name attributes and methods in such a way that it is obvious what their purpose are and how to use them. A programmer does not need to know how the actual presentation will look like, but it can help in the work of designing the code if there is a blue print to follow.

The programmer's perspective and its contents have been described in an earlier chapter, 'Custom Tag Libraries' in Chapter 8.1 under 'Separating the front and back end programming', Chapter 8. The main areas contain the Tag Library Descriptor file and the Tag Handlers.

9.2.2 Custom Tag Use cases

The following use cases describe the procedures taken place when creating a web page with custom tags and how to create the actual tags. The first is a client-side action, while the latter is a server-side action.

Task: Develop web page with custom tags

Participating Actors: Web developer

BASIC FLOW

Flow of Events:

1. Web designer starts designing the web page in for example Dreamweaver.
2. The tag is inserted
3. The web designer can see the tag in the web browser

ALTERNATIVE FLOW: TAG NOT DEFINED

Pre-conditions:

1. The tag is not defined

Flow of Events:

1. The web developer contacts the programmer

Task: Develop custom tag

Participating Actors: Programmer

BASIC FLOW

Flow of Events:

1. A tag library descriptor (TLD) file is created
2. The tag is defined in a handler class
3. The Java code for the tag is implemented

ALTERNATIVE FLOW: THE TLD IS ALREADY CREATED

Pre-conditions:

1. The TLD is already created

Flow of Events:

1. The programmer adds the tag to the TLD.

9.3 Master Design

The Tetra Pak filling machines can be monitored on one or more computers at the same time. The goal is that changes in the PLC should be shown as quickly as possible on all computers simultaneously. The technique making this possible is the push technology, developed by Tetra Pak.

If there are several computers to monitor one machine, still only one can be master. Being master means that the specific computer is capable of not only viewing data, but also changing data. For example all computers can see the alarms reported by the PLC, but only one (the master) can acknowledge them. All commands, apart from viewing, are non active for everyone except the master.

There should not be one computer that is master permanently, but rather the master role should rotate between the different computers. A computer that is currently not master can at any time claim the master role to become master. The only restriction is that you can not become master, if you are outside the Tetra Pak site or by other specified reasons are not entitled to become master. This is for security reasons, so that the alarms can not be acknowledged by someone without authority.
To sum up the requirements for the master design the system should fulfil the following:

R1. It should only be possible to acknowledge alarms if you possess the master role
R2. There should only be one master at a time
R3. It should be able to handle the situation if a computer which possesses the master role crashes
R4. It should be possible for every computer connected (on the Tetra Pak site) to become master
R5. A timeout (10 seconds) should prevent inactive computers holding the master role
R6. It should be possible to request the master role at any given time
R7. It should be possible to accept or decline a release of the master role
R8. In case of a rejection on a master request, the system should send a rejection message alert to the user who requested the master role
R9. It should be possible to access the web HMI from all trusted and authorized computers

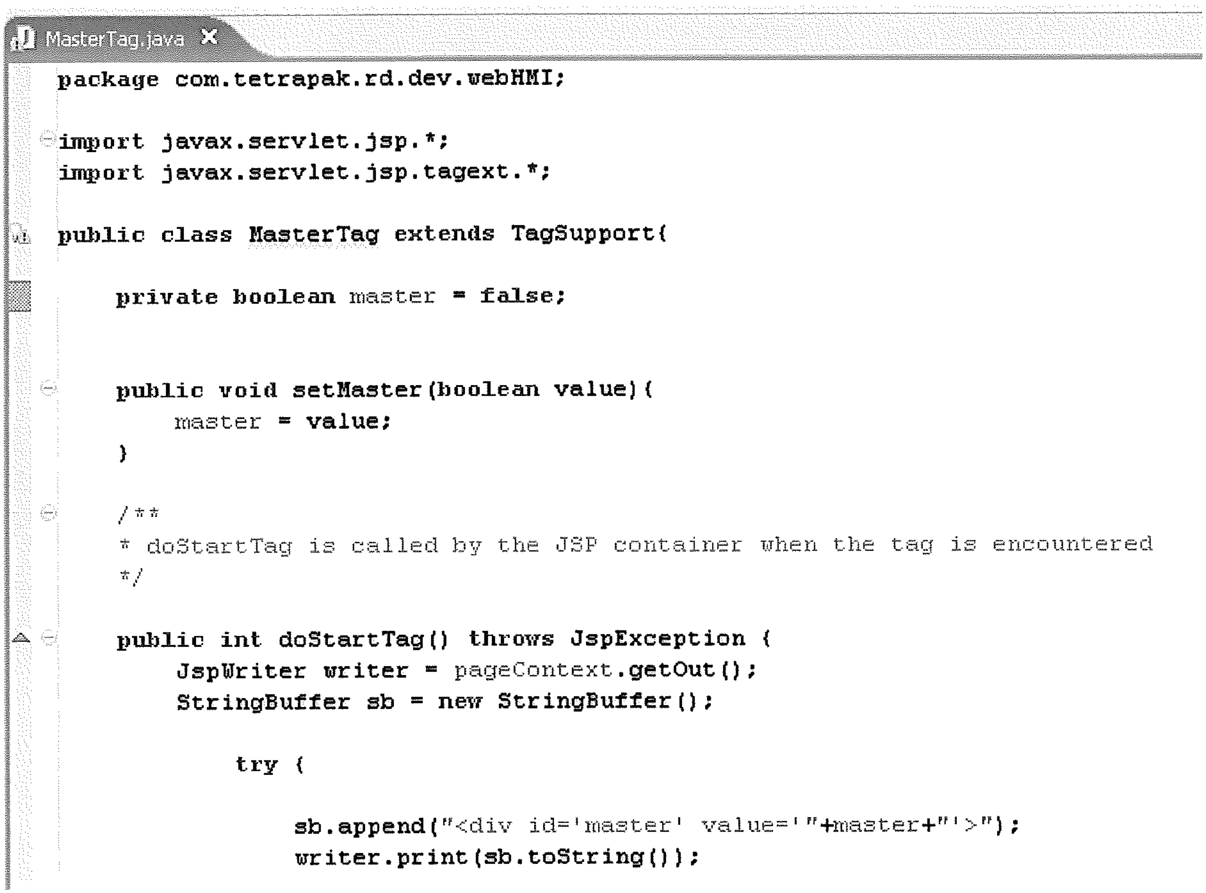
The results of the master design
A design that met all the specified requirements was implemented and tested. The solution is built upon a design where the server uses a session object to keep him updated about which client possesses the master role, but letting the clients handle the changes in the session object. This brings about several advantages, among others the fact that every asynchronous call is encrypted.

This solution requires a manual setting of the first master, but this can be done easily thanks to the usage of a master tag. In this tag, the web designer sets the attribute `master='true'`. When master rights are excluded for a client, the web designer can choose to not implement the master tag at all. Clients without authority will not be presented the interface where asking and releasing the master role is implemented.

The following code snippet shows the master tag in the *.tld-file.

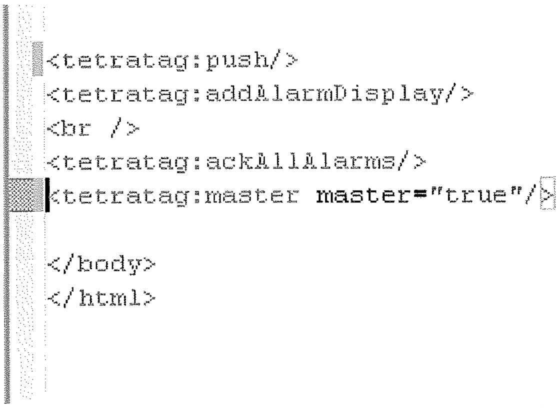
```
<!-- Tag to set master privilege. Initial value 'master' will always be
false -->

<tag>
     <name>master</name>
     <tagclass>com.tetrapak.rd.dev.webHMI.MasterTag</tagclass>
     <bodycontent>empty</bodycontent>
     <attribute>
          <name>master</name>
          <required>false</required>
     </attribute>
</tag>
```

In Figure 9.5 one can see how the initial value for master is set to 'false'. This is done in the MasterTag.java file accordingly to custom tag development described earlier. Figure 9.6 shows a web designer's view where the master tag's value can be changed to 'true'.



```
package com.tetrapak.rd.dev.webHMI;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class MasterTag extends TagSupport{

    private boolean master = false;


    public void setMaster(boolean value){
        master = value;
    }

    /**
    * doStartTag is called by the JSP container when the tag is encountered
    */

    public int doStartTag() throws JspException {
        JspWriter writer = pageContext.getOut();
        StringBuffer sb = new StringBuffer();

            try {

                sb.append("<div id='master' value='"+master+"'>");
                writer.print(sb.toString());
```

Figur 9.5 The initial value is set to false



```
<tetratag:push/>
<tetratag:addAlarmDisplay/>
<br />
<tetratag:ackAllAlarms/>
<tetratag:master master="true"/>

</body>
</html>
```

Figur 9.6 The web designer sets the attribute to true, giving the client master rights.

In a web based HMI not every computer needs to have the ability to interact with the web page. If a client only will possess viewing rights, rather than an administrative role, the web based solution needs to support this. That is why implementing the master interface as a custom tag was selected.

If a request for master is sent and the role is busy, for example in a situation where the holder is editing data and can not release the master role, a new request to become master must be sent. The resending request for the master role is not automatic, but must be done manually by the user.

The above described proceedings were implemented using the HttpSession object. As described in Chapter 7.1.4 this approach was the best suited for the output we wanted to accomplish. The architectural design worked well, and in Figure 9.7 we can see the basics of the session management.

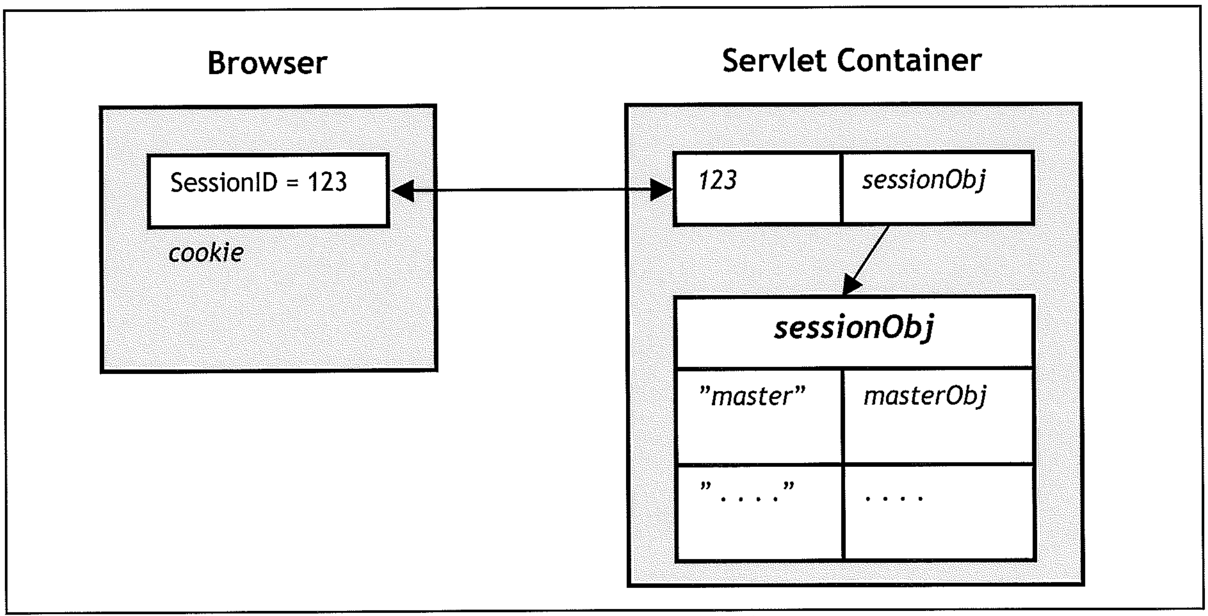


Figure 9.7 The general architecture of session management.

Further requirements concerning the way of indicating if a computer possesses the master role or not, must be dealt with by the web designer. These questions are outside the boundaries for this master thesis, but can still be interesting to have in mind while developing a suitable design for the master implementation. For example:

- What is a good way to indicate that one is the master?
- What is an appropriate way to inform the former master that he has lost access? An alert box?

10 Discussion

Separating business logic and design can be very beneficial for software development. Different areas of expertise can merge and create better design, not just in the general system architecture but also in the graphic presentation. On Tetra Pak's behalf several winnings can be achieved using this approach. Below some has been highlighted for further discussion.

10.1 Tetra Pak winnings

Tetra Pak has in the past always strived to be in the front row when it comes to development. The trend today is that many companies turn to the open source community [75]. A web based HMI is an opportunity for Tetra Pak to join this movement.

One of the advantages with using open source is the fact that there is a larger developer base for maintenance. Another way of looking at it is that there are more programmers in the labour pool, and that their competence might be higher than the IT-department within the company. If then the application as a whole is put out as an open source project, not only are there more people to test and report errors, there are also more programmers to fix them and to add new features.

Seen from a customer side of view using open source could also be interesting. If the customer experiences a problem he can directly post it to the community, instead of using the traditional way through Tetra Pak's technicians. This way the customer satisfaction will rise if the problem could be solved more rapidly.

There are several advantages with a web based HMI compared with the one in use. One of the largest is that a web based HMI give Tetra Pak the opportunity of being supplier independent. This means that it is possible for Tetra Pak to change supplier if the circumstances change. Among the suppliers this can also trigger a healthy competitiveness.

With this web based HMI the licence costs could be cut to a great extent. Everything used on this master thesis, except Dreamweaver, is at no cost for Tetra Pak.

10.2 Designing custom tag libraries

To achieve a good class library design we used a vocabulary derived from the problem domain. Tag and attribute names were chosen together with the Web page designer, which contributed to a mutual base from which we could build up our tag library. When our domain expertise was vague, several meetings with supervisor Mattias Wallinius and Web page designer Lama Ziegelwagner took place. Interviewing them helped us identifying which problems we needed to solve, but also created a mutual understanding for the terminology.

One of the problems was to really understand and separate what the developer/web designer would do. For example, there was a discussion where the data binding should take place. Would it be better to leave it to the programmer to decide for example which temperature that should trigger an alarm, or to the web designer?

As a whole, our goal was met and there is a small but working tag library to keep developing. It was satisfying for us to see the positive reaction from possible users and to hear their comments on improvement.

10.3 Performance

The performance issue of the application was quite complex to measure. One of the main reasons for this is the fact that we only developed a small number of tags compared to what is realistic for Tetra Pak to have. The graphic of the page was also simplified. Despite these facts the tests done indicate that there are a sufficient response time. Mattias Wallinius is part of the group having developed the solution described in Chapter 9.1, and according to them the users experienced the application as fast.

There was a problem with a memory leakage. The leakage was due to a problem with Firefox version 1.5.0.9. This can be avoided with tuning the settings in Firefox. Therefore it is only considered as a small obstruction.

Since there were no requirements set regarding performance at the start of the project, it is hard to determinate if performance is satisfying enough. Although not a formal requirement, it was always an ambition of ours to strive for an efficient application.

10.4 Summary

To sum this thesis up there are two specific areas of interest.

Fist, there has been a separation of the front and back-end programming. This has been done thanks to JSP and its ability to create tag libraries. As a result of this the programmer can focus on the content model and the web designer can work on the graphical representation.

There is also a new way to organise the responsibility of the monitoring computers, called Master Design. Only one computer can be master with the ability to interact with the PLC and the master role is rotating. The solution is built upon a design where the server uses a session object to keep him updated about which client possesses the master role, but letting the clients handle the changes in the session object. To solve the issue of the master responsibility a "master tag" was implemented. This gives the master rights that are excluded for a client. The web designer can choose to not implement the master tag at all.

11 Future work

During the inception phase different frameworks were evaluated, but due to time constraints these were never taken into actual use. This would, however, have a given place in further development. A framework facilitates the process when working in larger projects. The goal with this master thesis was to build the ground from which a complete web HMI could grow as an open source code project. To make this possible the interference of a framework is preferable.

A framework makes it possible to organize and develop a software project, making it possible to maintain a good structure throughout the whole development process. It supports developers to keep together the different components of a software project. A framework may include code libraries, a scripting language and/or other software whose purpose is to support during the development. There are several different types to choose among. Below is a short introduction of the different types and the advantages and disadvantages with each. Only open source frameworks have been considered.

- **Ajax framework**

  As described earlier Ajax stands for Asynchronous JavaScript and XML. It is a technology to build dynamic web pages on the client side, which enables page changes to be seen without refreshing the entire page. The benefits of this framework can be seen at two levels. The first one, on the client-side, where it offers JavaScript functions to send requests to the server and the second one, on the server-side, where it processes the requests, searches for the data and transmits them to the browser. This dividing eases the work of an Ajax programmer when building up web applications.[76]

- **Apache Struts**

  Apache Struts is a free open-source framework for creating Java web applications.[77] It is a Model-View-Controller (MVC) framework compatible with Sun's J2EE platform and primarily based on Servlet and JSP technology. [78]

  The advantages with this framework are many, among them the possibility for different groups of people to work with design and implementation of large web applications simultaneously. This enables a situation where single developers can concentrate on their own piece of work and do not have to think about how it will cooperate with the others. Apache Struts supports different presentation layers, there among JSP, XML and JavaServer Faces and also different model layers, for example JavaBeans and EJB. Another advantage is its popularity and the fact that it is very well documented.

  The disadvantage is that Struts has been around for quite a while and now is facing opposition in form of newer and more "light weight" MVC frameworks.

- **Tapestry**

  Tapestry is a rather new actor on the framework market which uses its Java-based programming toolkit to implement applications in accordance with MVC design pattern. Tapestry was developed with the user in focus, with simplicity and ease of use as two strong contributing factors. Tapestry makes it possible to create dynamic, robust and highly scalable web applications. It complements and builds upon the standard Java Servlet API, and so it works in any Servlet container or application server.[79]

Tapestry was created by Howard Lewis Ship and is now a part of the Apache Software Foundation. This means that Tapestry meet both quality and have sufficient support in both contributors and users.[80]

Appendix A Attributes for JSP commands

A.1 Attributes for <jsp:useBean>

Attribute	Description
id	Represents the object's identity within the specified range. It has to follow name conventions.
scope	Represents the object's range: page, request, session or application.
class	Represents the explicit class name which contains all package names and defines the object's implementation.
beanName	Represents the object's name. When *instantiate()* is called in java.beans.Beans, this name is used.
type	Represents the script variable's type. If not stated, the attribute is set to the same as in class.

A.2 Attributes for <jsp:setProperty>

Attribute	Description
name	Represents a name which has been declared in <jsp:useBean> or any other action.
property	Is used to specify the property. If set to * the action will iterate through the parameters from the request and match them with properties. If the parameter name matches the property name they will be set to the value of the parameter. If the parameter has an empty string as a value, the property will not be changed.
param	Represents the name of the parameter whose value will allot to the property.
value	Represents a value that will allot to the property.

* <jsp:setProperty> can not contain both a param and a value attribute.

A.3 Attributes for <jsp:getProperty>

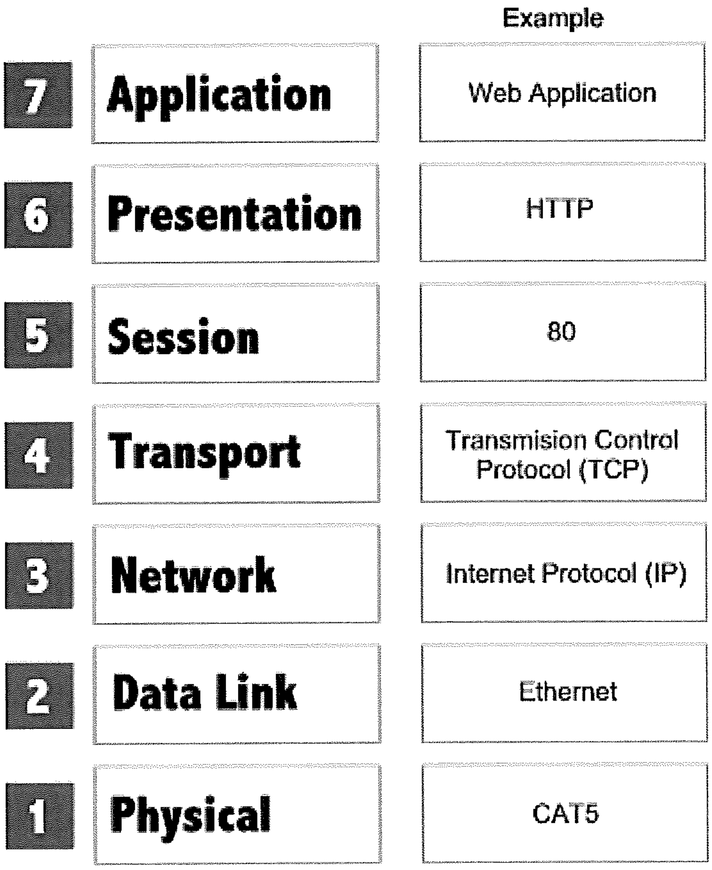
Attribute	Description
name	Represents a name which has been declared in <jsp:useBean> or any other action.
property	This attribute is used to state which property should be read within the given JavaBean.

A.4 Values of the attribute scope in the <jsp:useBean> action

Attribute	Description
page	Objects will be accessible only within the page where they were created. When the response is sent to the client or when the request is forwarded to another resource the reference will be released. Objects with the page range are stored in the pageContext object.
request	Objects with this range will be accessible within pages that handle the same request as the one where the object was created. When the request is finished the reference will be released. In case of forwarding to another page, objects will be accessible within the same runtime environment. Objects with the request range are stored in the request object.
session	Objects which are created in a session will be accessible in all pages within that session. When a session is over the resources are released. Objects with the session range are stored in the session object.
application	Objects are accessible in all pages which are worked up in the same application as the one where the object was created. Resources are released when the runtime environment return to ServletContext. Objects with the application range are stored in the application object that is associated with the page.

Appendix B OSI reference model

A schematic picture describing the different layers in the OSI reference model can be viewed below.[81]



		Example
7	Application	Web Application
6	Presentation	HTTP
5	Session	80
4	Transport	Transmision Control Protocol (TCP)
3	Network	Internet Protocol (IP)
2	Data Link	Ethernet
1	Physical	CAT5

Appendix C JavaServer Pages API Documentation

The following methods and constants were used for the tag functionality.[82]

doStartTag

```
public int doStartTag()
               throws JspException
```

Process the start tag for this instance. This method is invoked by the JSP page implementation object.

The doStartTag method assumes that the properties pageContext and parent have been set. It also assumes that any properties exposed as attributes have been set too. When this method is invoked, the body has not yet been evaluated.

This method returns Tag.**EVAL_BODY_INCLUDE** or BodyTag.**EVAL_BODY_BUFFERED** to indicate that the body of the action should be evaluated or **SKIP_BODY** to indicate otherwise.

When a Tag returns **EVAL_BODY_INCLUDE** the result of evaluating the body (if any) is included into the current "out" JspWriter as it happens and then doEndTag() is invoked.

BodyTag.EVAL_BODY_BUFFERED is only valid if the tag handler implements BodyTag.

The JSP container will resynchronize the values of any AT_BEGIN and NESTED variables (defined by the associated TagExtraInfo or TLD) after the invocation of doStartTag(), except for a tag handler implementing BodyTag whose doStartTag() method returns BodyTag.EVAL_BODY_BUFFERED.

Returns:

EVAL_BODY_INCLUDE if the tag wants to process body, SKIP_BODY if it does not want to process it.

Throws:

JspException - if an error occurred while processing this tag.

doEndTag

```
public int doEndTag()
             throws JspException
```

Process the end tag for this instance. This method is invoked by the JSP page implementation object on all Tag handlers.

This method will be called after returning from doStartTag. The body of the action may or may not have been evaluated, depending on the return value of doStartTag.

If this method returns EVAL_PAGE, the rest of the page continues to be evaluated. If this method returns SKIP_PAGE, the rest of the page is not evaluated, the request is completed, and the doEndTag() methods of enclosing tags are not invoked. If this request was forwarded or included from another page (or Servlet), only the current page evaluation is stopped.

The JSP container will resynchronize the values of any AT_BEGIN and AT_END variables (defined by the associated TagExtraInfo or TLD) after the invocation of doEndTag().

Returns:

indication of whether to continue evaluating the JSP page.

Throws:

JspException - if an error occurred while processing this tag

getOut

```
public abstract JspWriter getOut()
```

The current value of the out object (a JspWriter).

Returns:

the current JspWriter stream being used for client response

SKIP_BODY

```
public static final int SKIP_BODY
```

Skip body evaluation. Valid return value for doStartTag and doAfterBody.

EVAL_PAGE

```
public static final int EVAL_PAGE
```

Continue evaluating the page. Valid return value for doEndTag().

Appendix D HTTP request methods

HEAD

Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

GET

Requests a representation of the specified resource. By far the most common method used on the Web today. Should not be used for operations that cause side-effects (using it for actions in web applications is a common misuse). See 'safe methods' below.

POST

Submits data to be processed (e.g. from a HTML form) to the identified resource. The data is included in the body of the request.

PUT

Uploads a representation of the specified resource.

DELETE

Deletes the specified resource.

TRACE

Echoes back the received request, so that a client can see what intermediate servers are adding or changing in the request.

OPTIONS

Returns the HTTP methods that the server supports. This can be used to check the functionality of a web server.

CONNECT

For use with a proxy that can change to being an SSL tunnel.

Appendix E HttpSession Object

`isNew()`

Allows you to do determine whether the session is new. Note that a session is considered new if the server uses only cookie-based sessions, and the client has disabled the use of cookies. Otherwise, a session is considered not new when the client joins the session.

`getID ()`

Every HttpSession is assigned a unique identifier by the Servlet engine. The ID is extremely useful, as it can be used to put and get application layer data without fear of stepping on another user's data.

`setMaxinactiveInterval(int interval)`

`int getMaxInactiveInterval()`

This will allow you to set (and get) the maximum number of seconds that a session can remain inactive. If the time has expired, the session will be invalidated.

`getCreationTime()`

Allows you to get the time that the session was created. Returns the time in milliseconds since midnight January 1, 1970 GMT.

`getLastAccessedTime()`

Allows you to find out the last time the session was accessed. Measured in the same way as getCreationTime(), see above.

`invalidate()`

Calling this method will immediately invalidate the HttpSession and unbind any data in it.

Appendix F Essentials of JSP

When creating JSP pages predefined objects can be used. These are:[83]

- **`request (javax.servlet.http.HttpServletRequest)`**
  The interface makes it possible to get to the information in the HTTP protocol's header. Amongst the most commonly used areas for this object is to get parameters that belongs to a specific request. This is done by calling *getParameter()* with the wanted parameter as an argument. The method will return a string with the requested parameter's value.

- **`response (javax.servlet.http.HttpServletResponse)`**
  `response` represents the answer that will be sent back to the client. This object can be used to change the information in the HTTP protocol's header. It is mostly used to write HTML code that represents the result of calling the method and which is sent back to the client. Instead of *getWriter()* another predefined object can be used, namely *out.*

- **`pageContext`**
  This predefined object gives access to the JSP page's namespace and to other predefined objects. The instance which `pageContext` refer to is implementation dependent. It is created via a call to `JspFactory`. `pageContext` is used first and foremost in the generated code and it is not referred so often directly in the JSP page. Creating own tags is one important exception.

- **`session (javax.servlet.http.HttpSession)`**
  `session` is used for storing references to objects between the requests within one session. The variable is initiated via calling *getSession()* on the `pageContext` object in the Servlet that is generated. A code snippet shows the initiation:
  `session = pageContext.getSession();`

- **`application (javax.servlet.ServletContext)`**
  This variable gets a reference to the present ServletContext object via calling a pageContext method. Its range is set to application (see explanation on different range possibilities on page XX), which makes it accessible for all JSP pages as long as the server is up. It is mostly used for gaining access to information about the surrounding environment. Information is stored in the ServletContext object which is a preferable place due to the advantage to share information between different JSP pages and Servlets. application is therefore used for both storage and accessing information.

- **`out`**
  `out` is an instance of the `JspWriter` class which is a subclass to `java.io.Writer`. It is used for writing the result back to the client. The argument is HTML code which is shown in the client's web browser (see response).

- **`config`**

This object is generated by the server and contains the server's configuration. Access to configuration information is already available via the application object why `config` normally can be excluded.

- **`page`**
  The `page` object is a reference to the current instance of the JSP page and is set to this in the generated Servlet.

- **`exception`**
  This object only exists in pages which are declared to be error pages. The variable is a reference to the exception that caused the request to the error page.

Appendix G Applet solution

Push Applet

In this section a brief explanation about the Applet solution will follow after an introduction of what defines an Applet. Performance and security issues are also mentioned.

What is an Applet?

In contrary from Servlets, Applets are small applications which are executed in the browser on the client side. To be able to run an applet, browsers need to be enabled with Java technology and the program needs to be compiled. In other words, an applet is a class file which is called by an element in HTML when needed. Other class files and/or whole class libraries can be needed for the applet to function. Further on an applet must be a subclass of the `java.applet.Applet` class, which provides the standard interface between the applet and the browser environment.[84]

Applets do not have a main method that is automatically called to begin the program. Instead, several methods are called at different points in the execution of an applet. When an applet is loaded in a web page, the browser manages the applet's life cycle which basically contains of four methods.

- *init()* - This method initialize the applet. This can only be done once and only once.
- *start()* - This method is automatically called after the init()-method but also when a user returns to a page containing the applet after visiting other pages. This can be done one or more times during an applets lifecycle.
- *stop()* - This method is automatically called when a user leaves the page where the applet is running, but can also be used to stop an animation. This can also be done one or more times.
- *destroy()* - This method is only called when the browser shuts down normally. An applet can only be destroyed once and only once.

An applet such as this is typically managed and run by Java Plug-in. Java Plug-in, which is automatically included when you download the Java(TM) SE Runtime Environment (JRE), extends the functionality of a web browser, allowing applets to run under Sun's Java(TM) SE Runtime Environment (JRE) rather than the Java Runtime Environment that comes with the web browser. It works with the Mozilla family of browsers and with Internet Explorer.[85]

Security

For security reasons, applets that are loaded over the network have several restrictions. One is that an applet can not ordinarily read or write files on the computer that it is executing on. Another is that an applet can not make network connections except to the host that it came from. Despite these restrictions, applets can do some things that you might not expect. For example, applets can invoke the public methods of other applets on the same page.

Proxy

A proxy is a server that exists between the web browser and the server. During the communication phase the proxy intercepts all requests that the client sends to the server to see if it can fulfil them itself. If the request can not be taken care of, it is forwarded to the real receiver. Proxy servers have two main purposes. The first is its ability to improve performance for different user on the Web. By saving requests for a certain amount of time it can return an already fetched page without retrieving it from the hosting server again. This saves both time and traffic on the network. The other purpose is to filter requests. When preventing users from gaining access to certain web sites a proxy server comes in handy.[86]

Appendix H JavaBean actions

There are three defined standard actions that facilitate JavaBean usage.[87]

`<jsp:useBean>`

This action associates a script variable with a certain ID with an instance of a JavaBean with the same ID and a specified range. It is flexible in the behaviour of trying to find the matching object; if it fails it tries to create a new object. It is also possible to use this action to rename an already declared object locally. The syntax is as follows:

```
<jsp:useBean id=″name″ scope=″page|request|session|application″ typeSpec />
```

`<jsp:setProperty>`

This action sets a property of a JavaBean. The name attribute has to contain the name of an object which is defined within current range. The syntax is as follows:

```
<jsp:setProperty name=″beanName″ prop_expr />
```

`<jsp:getProperty>`

This action takes the value from a specified property, converts it to a string and places it in the out object. The stated instance of the JavaBean has to be defined within the current range. The syntax is as follows:

```
<jsp:getProperty name=″name″ property=″propertyName″ />
```

Appendix I What is WEB 2.0?

WEB 2.0 is about many different phenomenons. It is a new way of thinking where the creator of a web page is no longer the one who controls the information displayed, if anything it is now the visitor of the page. Here Google, Blogs and different torrent pages can be mentioned as examples. WEB 2.0 is about the ability to avoid limitations and see the web as a platform, where it does not matter which technique is being used. Below other definitions of WEB 2.0 are stated:[88]

- The Web should be transparent where any device could be used to access the Web.
- The user creates the information on the web page.
- Freedom of choice eliminates the former way of locking in clients.
- Portal thinking facilitates the creation of well equipped portals where clients have the freedom to choose and select what she wants.
- Companies' ability to think in other terms where a change in their business model keeps market shares up.

The creation of new standards or proposals to new standards is managed by World Wide Web Consortium. W3C is an organisation for standardization.[89] Their guidelines facilitate for both software and hardware companies to strive for compatible technology. To read more, please visit their homepage at http://www.w3c.org.

References

[1] Metsker, S., J., Wake, C. W., Design Patterns in Java, Pearson Education Limited, England, 2006
[2] World Wide Web Consortium, http://www.w3.org/
[3] PHP:Hypertext preprocessor. http://www.php.net/
[4] Sun Opens Java. http://www.sun.com/2006-1113/feature/story.jsp
[5] A Manager's Introduction toThe Rational Unified Process (RUP) http://www.ambysoft.com/dow/managersIntroToRUP.pdf
[6] Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Rational_Unified_Process
[7] Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Rational_Unified_Process
[8] A Manager's Introduction toThe Rational Unified Process (RUP) http://www.ambysoft.com/dow/managersIntroToRUP.pdf
[9] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Rational_Unified_Process
[10] A Manager's Introduction toThe Rational Unified Process (RUP) .http://www.ambysoft.com/dow/managersIntroToRUP.pdf
[11] A Manager's Introduction toThe Rational Unified Process (RUP). http://www.ambysoft.com/dow/managersIntroToRUP.pdf
[12] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Rational_Unified_Process
[13] Application Development Trends. http://www.adtmag.com/article.aspx?id=18913
[14] Manifesto for Agile Software Development http://agilemanifesto.org/
[15] Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Agile_software_development
[16] SCRUM, in five minutes. http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf
[17] SoftwareConfigurationManagement http://www.cmcrossroads.com/cgi-bin/cmwiki/view/CM/SoftwareConfigurationManagement
[18] Subversion. http://subversion.tigris.org/
[19] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Rational_Unified_Process
[20] A Manager's Introduction toThe Rational Unified Process (RUP) http://www.ambysoft.com/dow/managersIntroToRUP.pdf
[21] Jonas Webresurs. http://www.jonasweb.nu
[22] Introduction to HTML. http://www.w3schools.com/HTML/HTML_intro.asp
[23] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/HTML
[24] Introduction to HTML 4. http://www.w3.org/TR/HTML4/intro/intro.HTML
[25] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/XHTML
[26] Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/SGML
[27]Extensible Markup Language (XML). http://www.w3.org/TR/REC-xml/
[28] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Cascading_Style_Sheets
[29] W3C Technical Reports and Publications. http://www.w3.org/TR/
[30] JavaScript Introduction. http://www.w3schools.com/js/js_intro.asp
[31] About JavaScript. http://developer.mozilla.org/en/docs/About_JavaScript
[32] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/JavaScript
[33] Flanagan, David, JavaScript- The Definitive Guide, O´Reilly &Associates, Inc, USA, 1998
[34] XML i 10 punkter http://www.w3c.se/resources/office/translations/XML-in-10-points_sw.html
[35] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/DHTML
[36] Introduction to DHTML. http://www.w3schools.com/dhtml/dhtml_intro.asp
[37] Adobe Flash, http://www.adobe.com/products/flash/flashpro/

[38] Do I Need DHTML? http://www.netmechanic.com/news/vol3/beginner_no14.htm
[39] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Apache_Tomcat
[40] Merck & Co., Inc. is a global research-driven pharmaceutical company. http://www.merck.com/about/
[41] Tomcat FAQ- FDA. http://tomcat.apache.org/faq/fda-validation.html
[42] Tomcat FAQ- Security http://tomcat.apache.org/faq/security.html
[43] Using OpenSSL to set up your own CA. http://marc.theaimsgroup.com/?l=tomcat-user&m=106293430225790&w=2
[44] Apache Tomcat. http://tomcat.apache.org/
[45] What Is a Servlet? http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets2.html
[46] Advanced Servlets and Java Server Pages[tm] Programming. http://developers.sun.com/events/techdays/presentations/dchen/pdf/dchen1a.pdf
[47] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Servlet
[48] Chopra, V., Balore, A., Eaves, J., Galbraith, B., Li, S., Wiggers, C., Apache Tomcat 5, Wiley Publishing, Indianapolis, Indiana, 2004
[49] ONJava.com- Designing JSP Custom Tag Libraries. http://www.onjava.com/pub/a/onjava/2000/12/15/jsp_custom_tags.html
[50] JSP best practices: Combine JavaBeans components and JSP technology. http://www-128.ibm.com/developerworks/java/library/j-jsp05133.html
[51] JavaBeans FAQ: General Questions. http://java.sun.com/products/javabeans/faq/faq.general.html
[52] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Javabean
[53] Ballard, Phil, Kom igång med Ajax, Pagina, Sweden, 2006
[54] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/AJAX
[55] Distributed Computation with Java Remote Method Invocation. http://www.acm.org/crossroads/xrds6-5/ovp65.html
[56] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/HTTP
[57] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/HTTP
[58] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Http#Safe_methods
[59] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Https
[60] Sierra, K., Bates, B., Basham, B., Head First Servlets & JSP, O'Reilly Media, USA, 2004
[61] Sun Developer Network, FAQ, http://java.sun.com/blueprints/qanda/web_tier/session_state.html
[62] Tag Libraries Tutorial. http://java.sun.com/products/jsp/tutorial/TagLibraries3.html#63159
[63] ONJava.com- Designing JSP Custom Tag Libraries. http://www.onjava.com/pub/a/onjava/2000/12/15/jsp_custom_tags.html
[64] Metsker, S., J., Wake, C. W., Design Patterns in Java, Pearson Education Limited, England, 2006
[65] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Model-view-controller
[66] Metsker, S., J., Wake, C. W., Design Patterns in Java, Pearson Education Limited, England, 2006
[67] Koskimies, Kai. "Removing Dependencies". http://www.cs.tut.fi/~kk
[68] Black, Steven, "Design Pattern: Mediator". http://www.stevenblack.com/Articles/PTN-Mediator.ASP

[69] The Mediator (Behavioral) Design Pattern by Gopalan Suresh Raj. http://my.execpc.com/~gopalan/design/behavioral/mediator/mediator.html
[70] Metsker, S., J., Wake, C. W., Design Patterns in Java, Pearson Education Limited, England, 2006
[71] Using JUnit With Eclipse IDE. http://www.onjava.com/pub/a/onjava/2004/02/04/juie.html
[72] Eclipse Junit testing tutorial. http://www.laliluna.de/eclipse-junit-testing-tutorial.html
[73] Apache JMeter. http://jakarta.apache.org/jmeter/
[74] JMeter tips. http://www.javaworld.com/javaworld/jw-07-2005/jw-0711-jmeter.html?page=3
[75] IT Manager's Journal. http://www.itmanagersjournal.com/feature/196
[76] Wikipedia, the free encyclopedia.http://en.wikipedia.org/wiki/Ajax_framework
[77] Apache Struts. http://struts.apache.org/
[78] ONJava.com - Introduction to Jakarta Struts Framework http://www.onjava.com/pub/a/onjava/2001/09/11/jsp_servlets.HTML
[79] Tapestry. http://tapestry.apache.org
[80] Tapestry is now Apache Tapestry. http://www.theserverside.com/news/thread.tss?thread_id=41132
[81] Learnthat.com- free Tutorials and free courses http://www.learnthat.com
[82] Servlet and JavaServer Pages API Documentation. http://tomcat.apache.org/tomcat-4.1-doc/servletapi/
[83] Goodwill, James, JSP-Java Server Pages, Pearson Education Limited, England, 2001
[84] Grundläggande Java - del 9 http://iwtjanster.idg.se/webbstudio/pub/artikel.asp?id=194
[85] Applets. http://java.sun.com/applets/
[86] Proxy server. http://www.webopedia.com/TERM/p/proxy_server.html
[87] Goodwill, James, JSP-Java Server Pages, Pearson Education Limited, England, 2001
[88] Jonas Webresurs. http://www.jonasweb.nu
[89] World Wide Web Consortium. http://www.w3.org/