

Date of acceptance Grade

Instructor

Software design of a dynamic and data intensive Web server

Alparslan Ünsal

Helsinki November 20, 2007
Seminar Report
UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet/Sektion – Faculty/Section Faculty of Science		Laitos – Institution – Department Computer Science	
Tekijä – Författare – Author Alparslan Ünsal			
Työn nimi – Arbetets titel – Title Software design of a dynamic and data intensive Web server			
Oppiaine – Läroämne – Subject Computer Science			
Työn laji – Arbetets art – Level Seminar report	Aika – Datum – Month and year 20.11.2007	Sivumäärä – Sidoantal – Number of pages 17 pages	
Tiivistelmä – Referat – Abstract <p>This seminar report is a part of my pro gradu thesis. My pro gradu thesis is about designing a dynamic and data intensive Web server. The design work will be done for PIEngeering Ltd.</p> <p>In this seminar report some information is given about web applications, software application frameworks, Web application frameworks and different Web application frameworks. I believe that this background knowledge will be beneficial for my pro gradu thesis.</p> <p>ACM Computing Classification System (CCS):</p> <p>C.2 Computer-Communication Networks C.2.4 Distributed Systems</p> <ul style="list-style-type: none"> • Client/server • Distributed applications <p>D.2 Software Engineering D.2.10 Design Methodologies</p> <ul style="list-style-type: none"> • Methodologies • Representation 			
Avainsanat – Nyckelord – Keywords Software design, server design, web application framework			
Säilytyspaikka – Förvaringställe – Where deposited			
Muita tietoja – Övriga uppgifter – Additional information			

Contents

1	Introduction	1
2	Web application frameworks	2
2.1	Web applications	2
2.2	Software application frameworks	3
2.3	Web application frameworks	4
2.3.1	Apache struts	6
2.3.2	DotNetNuke	7
2.3.3	Ruby on rails	8
2.3.4	Spring framework	9
2.3.5	Wicket	10
2.3.6	JSF	10
2.3.7	Tapestry	11
2.3.8	Zope	12
3	Modelling techniques of Web applications with UML	12
4	Security in Web applications	13
4.1	Security architectures	13
5	The problem domain and the software process	13
5.1	The problem domain and requirements	13
5.2	Software process	13
6	Implementation of design	14
6.1	Static diagrams	14
6.2	Dynamic diagrams	14
7	Results	14
8	Summary	14
	References	15

1 Introduction

My pro gradu thesis is about designing a dynamic and data intensive Web server. The design work will be done for PIEnengineering Ltd.

The design phase is one of the most important phases of a software project, because software defects are mostly found in the requirements and design phase [BMU75]. The design phase involves several activities: architectural design, subsystem design, component design, class design, data structures design, algorithm design and user interface design (Figure 1).

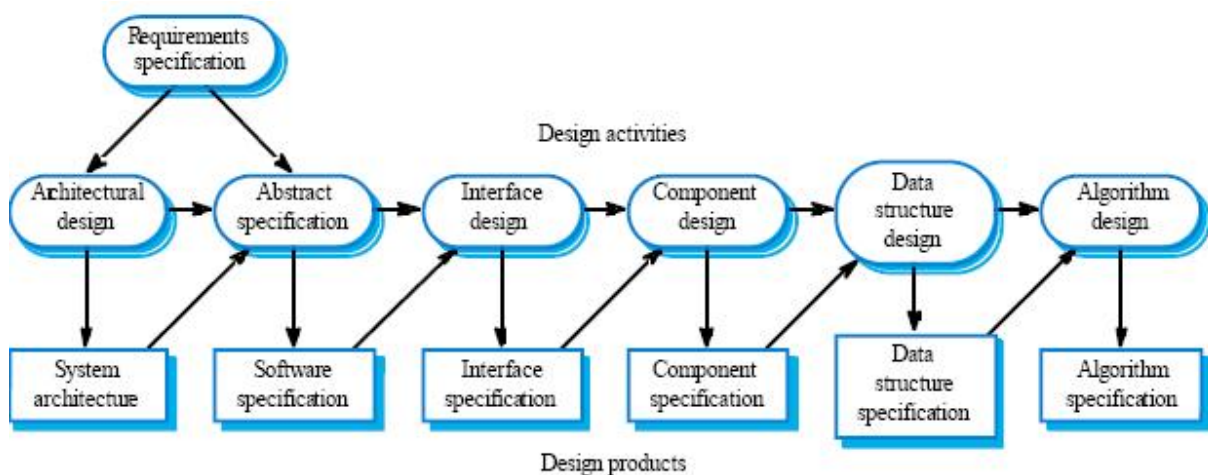


Figure 1. Design activities and products [Som04]

A decision has to be made before architectural design, whether a Web application framework will be used or not. I believe that a Web application framework will be used for the Web application, because Web application frameworks significantly reduce the amount of time, effort, and resources required to develop and maintain Web applications [ShH06]. In this seminar report some information is given about web applications, software application frameworks, Web application frameworks and different Web application frameworks. I believe that this background knowledge will be beneficial for my pro gradu thesis.

2 Web application frameworks

Before going to explain the Web application frameworks, a definition about Web applications and software frameworks should be given.

2.1 Web applications

A Web application is an application that is accessed via Web over a network such as the Internet or an intranet [Wik07a].

In earlier types of client-server computing, each application had its own client program which served as its user interface and had to be separately installed on each user's personal computer. An upgrade to the server part of the application would typically require an upgrade to the clients installed on each user workstation, adding to the support cost and decreasing productivity. In contrast, Web applications dynamically generate a series of Web documents in a standard format supported by common browsers such as HTML/XHTML [Wik07a].

Some pages include client side scripts that are interpreted by the browser. These scripts define additional dynamic behavior for the display page and often interact with the browser, page content and additional controls (Applets, ActiveX controls and plug-ins) contained in the page [LCC00].

A Web application is commonly structured as a three-tiered application (Figure 2). In its most common form, a Web browser is the first tier, an engine using some dynamic Web content technology (such as ASP, ASP.NET, CGI, ColdFusion, JSP/Java, PHP, Python, or Ruby On Rails) is the middle tier, and a database is the third tier. The Web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface [Wik07a].

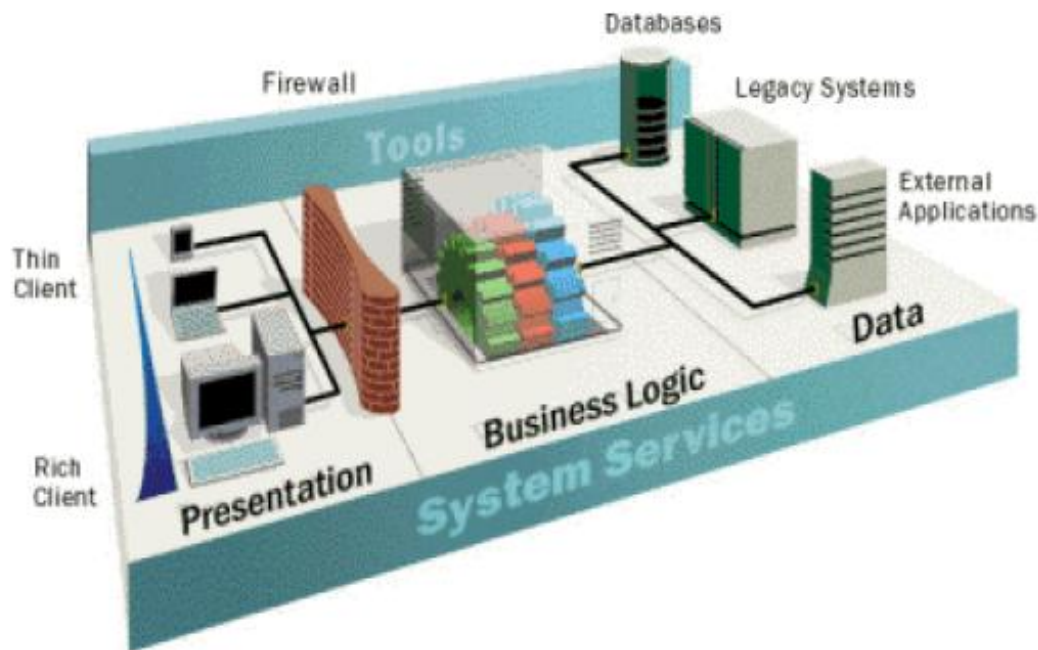


Figure 2. n-tier architecture of a Web application [Sep02]

The business logic layer consists of some business objects that execute the business logic by their collaboration. In the runtime, the Web server calls the interface of business object, which has access to all the information that came along with the page request (values of form fields and parameters). The business object uses the request details, and typically accesses server side resources to produce a response to client. Different functionality can be obtained by passing parameters to the business object [LCC00].

2.2 Software application frameworks

In information systems environment, a framework is a defined support structure in which other software applications can be organized and developed. A framework may include support programs, code libraries, a scripting language, common services, interfaces, or other software packages/utilities to help develop and glue together the different components of a software application. A software framework is a reusable design and building blocks for a software system and/or subsystem [ShH06].

A software framework consists of frozen spots and hot spots [Pre94]. The frozen spots define the overall architecture of a software system – its basic components and the relationships between them. These remain unchanged (frozen) in any instantiation of the application framework. On the other hand, hot spots represent those parts of the software framework that are specific to individual software systems. Hot spots are designed to be generic. In other words, they can be adapted to the needs of the application under development [ShH06].

In an object-oriented environment, a framework consists of abstract and concrete classes. Instantiation of such a framework consists of composing and subclassing the existing classes [Bus96].

2.3 Web application frameworks

A Web application framework is a software framework that is designed to support the development of dynamic websites, Web applications and Web services. The framework aims to alleviate the overhead associated with common activities used in Web development. For example, many frameworks provide libraries for database access, templating frameworks and session management, and often promote code reuse [Wik07b]. There are different kinds of Web application frameworks, which are used widely in Web applications. For example: Struts, Ruby on Rails, DotNetNuke, JBoss.

The Web Application Framework domain layer usually models basic concepts such as users, groups, and permissions. Web Application Frameworks may also include other relevant parts such as a user interface (UI) framework, a UI component library designed for the rapid development and reuse of Web user interfaces, and a powerful object-relational persistence engine/utility [ShH06].

Web Application Frameworks usually implement the Model-View-Controller (MVC) design pattern, typically in the Model 2 architecture to develop request-response Web-based applications on the Java EE and .Net models. It also integrates services such as

search, versioning, and permissions into the basic business objects, enabling applications to leverage framework services with little or no extra work [ShH06].

Reasons for using Web application frameworks

The software frameworks significantly reduce the amount of time, effort, and resources required to develop and maintain Web applications. Moreover, a framework is an open architecture based on commonly accepted standards (e.g., Java, .Net, XML, XSLT, JAAS, Servlet, JSP, JDBC, ADO.Net) and technologies (e.g., JUnit, XUnit, Ant, Log4j, JDom, Xalan, Xerces, Lucene), enabling any experienced developer to rapidly develop and support the system without a steep learning curve. This best-of-breed approach to adopting and integrating technologies enables application designers to focus on solving their business problems [ShH06].

The use of Web application frameworks can often reduce the number of errors in a program, both by making the code more simple, and by allowing one team to concentrate just on the framework [Wik07a].

A Web Application Framework is typically deployed in an n-tier architecture and uses proven, standard technology. By using standard technologies, a framework can be easily deployed within existing enterprise infrastructures, leveraging existing hardware, software, processes, and people [ShH06].

Virtually all Web applications have a common set of basic requirements, such as user management (e.g., secure user login, password recovery), group management, and access authorization. A Web Application Framework usually includes all these functionalities, refined through hundreds of production deployments, freeing developers to focus on the needs of their specific application [ShH06].

In addition to a basic set of services, Web applications typically have two other important similarities: they store important data in a relational database and they interact with users via a Web-based user interface. A sophisticated object-relational persistence layer

automatically manages how model objects are stored in the database. The persistence layer, by generating optimized SQL from metadata, drastically reduces the amount of effort required to model and refactor database schemas or support additional database architectures [ShH06].

A Web Application Framework may also include a component-based presentation rendering framework that enables developers to extend existing UI components or build new components that can be reused throughout an application, e.g. XSL and tag libraries. A breadth of domain services such as data validation, versioning, categorization, print, page navigation and full-text search may be provided. Any application written on top of a Web Application Framework can transparently and immediately take advantage of these basic services [ShH06].

There are many Web application frameworks and one of them must be chosen, in order to continue the design work. Some basic knowledge about the Web application frameworks is given in the following text.

2.3.1 Apache struts

Apache Struts is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture [Wik07c]. Struts was originally developed by Craig McClanahan and donated to the Apache Foundation in May 2000. Struts has been a de facto framework with a strong and vibrant user community [ShH06].

Struts uses and extends the Java Servlet API to adopt the "Model 2" approach, a variation of the classic Model-View-Controller (MVC) design pattern. Under Model 2, a Servlet (or equivalent) manages business logic execution, and presentation logic resides mainly in server pages [ShH06].

In a standard Java EE Web application, the client will typically submit information to the server via a Web form. The information is then either handed over to a Java Servlet which processes it, interacts with a database and produces an HTML-formatted response, or it is given to a JavaServer Pages (JSP) document which intermingles HTML and Java code to achieve the same result. Both approaches are often considered inadequate for large projects because they mix application logic with presentation and make maintenance difficult [Wik07c].

The goal of Struts is to cleanly separate the model (application logic that interacts with a database) from the view (HTML pages presented to the client) and the controller (instance that passes information between view and model). Struts provides the controller (a servlet known as ActionServlet) and facilitates the writing of templates for the view or presentation layer (typically in JSP, but XML/XSLT and Velocity are also supported). The Web application programmer is responsible for writing the model code, and for creating a central configuration file `struts-config.xml` which binds together model, view and controller [Wik07c].

Struts also supports i18n (internationalization), provides facilities for the validation of data submitted by Web forms, and includes a template mechanism called "Tiles" which (for instance) allows the presentation layer to be composed from independent header, footer, and content components [Wik07c].

Although Struts is a well documented, mature and popular framework for building front ends to Java applications, it is facing new challenges from newer "light weight" MVC frameworks such as Spring MVC, Stripes and Tapestry. [Wik07c].

2.3.2 DotNetNuke

The DotNetNuke application originally evolved out of another project, called the IBuySpy Workshop. The IBuySpy Workshop application had been created by Shaun Walker as an enhancement to the IBuySpy Portal starter kit. Microsoft had earlier released the IBuySpy Portal as a sample application for the .NET Framework [Wik07d].

DotNetNuke has a basic core that can be extended using pluggable modules and providers with additional functionality. The look and feel of individual sites can be customized using skins, which provides a clear separation between design and content, enabling a Web designer to develop skins without requiring any specialist knowledge of development in ASP.NET: only knowledge of HTML and an understanding of how to prepare and package the skins themselves are required [ShH06].

About a dozen basic modules are included with the core DotNetNuke distribution, and further modules can be downloaded from the DotNetNuke website, including e-commerce systems, photo galleries, blogs, forums, wiki and mailing list options. Additional third party modules are provided by both the open source community and proprietary commercial developers [Wik07d].

2.3.3 Ruby on rails

Ruby on Rails (RoR) is a free Web application framework. It aims to increase the speed and ease with which database-driven Web sites can be created, and offers skeleton code frameworks (scaffolding) from the outset. RoR is an open source project written in the Ruby programming language, and applications using the Rails framework are developed using the Model-View-Controller design paradigm [Wik07e].

Some of RoR's guiding principles are: *less software, convention over configuration, don't repeat yourself, scaffolding and immediate feedback.*

Less software means writing fewer lines of code to implement the application. Keeping the code small means faster development and fewer bugs, which makes the code easier to understand, maintain, and enhance [Hib05].

Convention over configuration means an end to verbose XML configuration files [Hib05]. RoR avoids configurations in their broadest sense. The only configuration file is called `database.yml`; it contains the type, user name, and password for an application database [Bäk07].

The *Don't Repeat Yourself* (DRY) principle requires every piece of system knowledge to have a unique, definite, and relevant representation. This approach means developers repeat themselves seldom or never. In this context, a piece of knowledge can be data or metadata, logic, functionality, or an algorithm. As an environment, RoR is designed to make singular declarations and reuse as easy as possible. The DRY principle is striking, for example, in Active Record, which determines attributes and class values exclusively in the database and not in the program code [Bäk07].

The database scheme and the *scaffold* console command let RoR create a basic skeleton of controllers and view templates that have create-retrieve update- delete (CRUD) functionality. In many programming situations, RoR also offers generators that save a lot of time for recurring elements, such as a login form. The automatic creation of project scaffolding saves time and lets the developer immediately start work on the application's core functionality [Bäk07].

Immediate feedback means that Developers can get an up-to-date status of their program by reloading it in the browser. No additional time is involved in deploying the program. RoR doesn't require projects to compile and execute first [Bäk07].

2.3.4 Spring framework

Spring is a layered Java EE application framework, which includes a lightweight container for automated configuration and wiring of application objects via inversion of control (aka dependency injection), an abstraction layer for transaction management, a JDBC abstraction layer, AOP functionality, and integration with O-R mappers. The origins of Spring can be traced back to a book by Rod Johnson, who presented his interface 21 framework that was later released into the open source world, this framework formed the foundation of the Spring framework. The first official 1.0 release was available in March 2004 [ShH06].

Although the Spring Framework does not enforce any specific programming model it has become widely popular in the Java community primarily as an alternative and replacement for the Enterprise JavaBean model. By design, the framework offers a lot of freedom to Java developers yet provides well-documented and easy to use solutions for common practices in the industry [Wik07f].

2.3.5 Wicket

Wicket is a framework that takes simplicity, separation of concerns and ease of development to a new level. Wicket pages can be mocked up, previewed and later revised using standard WYSIWYG HTML design tools. Dynamic content processing and form handling is all handled in Java code using a component model backed by POJO data beans that can easily be persisted using different technology. Wicket has a transparent state management with no XML configuration files. The first release of Wicket 1.0 went public in June 2005 [ShH06].

Wicket does not introduce any special syntax to HTML. Instead, it extends HTML in a standards-compliant way via a Wicket namespace that is fully compliant with the XHTML standard. This means that any HTML editor can be used to work on web pages and Wicket components [Apa07].

Web designers can work on the HTML with very little knowledge of the application code. Likewise, coders can work on the Java components that attach to the HTML without concerning themselves with what a given page looks like [Apa07].

2.3.6 JSF

JavaServer Faces (JSF) is a server-side user interface component framework for Java-based Web applications. JSF contains an API for representing UI components and managing their state; handling events, server-side validation, and data conversion;

defining page navigation; supporting internationalization and accessibility; and providing extensibility for all these features. It also contains two JSP (JavaServer Pages) custom tag libraries for expressing UI components within a JSP page and for wiring components to server-side objects. The specification of JSF 1.0 (JSR-127) was initially released in March 2004. JSF 1.2 Specification (JSR- 252) is the next generation of JSF, which has a final draft released in August 2005 [ShH06].

2.3.7 Tapestry

Tapestry is an object-oriented Java Web application framework to implement applications in accordance with the model-view-controller design pattern. Tapestry emphasizes simplicity, ease of use, and aims to avoid forcing programmers to create enormous blocks of "glue code". Tapestry uses a modular approach to Web development, by having strong binding between user interface components (objects) on the Web page and their corresponding Java classes. This component-based architecture borrows many ideas from WebObjects [Wik07g].

Tapestry complements and builds upon the standard Java Servlet API, and divides a Web application into a set of pages, each constructed from components. This provides a consistent structure, allowing the Tapestry framework to assume responsibility for key concerns such as URL construction and dispatch, persistent state storage on the client or on the server, user input validation, localization/internationalization, and exception reporting. Developing Tapestry applications involves creating HTML templates using plain HTML, and combining the templates with small amounts of Java code using (optional) XML descriptor files. In Tapestry, an application is created in terms of objects, and the methods and properties of those objects – and specifically not in terms of URLs and query parameters. Tapestry brings true object-oriented development to Java Web applications. Tapestry was originally created by Howard Lewis Ship, and the project was moved to the Apache Foundation around early 2004 [ShH06].

2.3.8 Zope

is an open-source, object-oriented Web application server written in the Python programming language. It can be almost fully managed with a Web-based user interface. Zope publishes on the Web Python objects that are typically persisted in an object database, ZODB. Basic object types, such as documents, images, and page templates, are available for the user to create and manage through the Web. Specialized object types, such as wikis, blogs, and photo galleries, are available as third-party add-ons (called products) [Wik07h].

A Zope website is composed of objects in an object database as opposed to files, as is usual with many other web server systems. This approach allows users to harness the advantages of object technologies, such as encapsulation. Zope maps URLs to objects using the containment hierarchy of such objects; methods are considered to be contained in their objects as well [Wik07h].

There are two major generations of the software in use today. As of July 2007, Zope 2.10.4 is the latest stable release of Zope 2 codebase, and Zope 3.3.1 is the latest release of Zope 3. Zope is distributed under the terms of the Zope Public License, a free software license [Wik07h].

3 Modelling techniques of Web applications with UML

Research has shown the importance of having an architecture document during the development of a software system [PeW92, ShG96]. Such a document improves developers' system understanding. It provides a building plan for the system and reduces its maintenance cost. It provides an overview description of the system. It permits the developer to view the major subsystems in the software system and the relations between them. Unfortunately, many software systems do not have an architecture document. The cost of manually developing this document increases with the size and the complexity of the software system [HaH00].

When diverse technologies are applied to a Web application, software engineers need to understand how components using the technologies are interrelated in terms of software architecture. In order to explain software architecture, a visual modeling technique has been used in software industry [ChL02].

This chapter is not complete yet, I will write it during my gradu thesis. There are already several papers written in this field. I will use the modelling techniques according to the papers. If needed, I will use the UML extension mechanisms to model new web objects.

4 Security in Web applications

This chapter will explain the basics of security in web applications and security architectures. For example: authentication, authorization, data encryption and so on.

4.1 Security architectures

5 The problem domain and the software process

I will try to explain the problem domain and show the gathered requirements. I will give information on the software process used.

5.1 The problem domain and requirements

5.2 Software process

6 Implementation of design

In this chapter I will show both static and dynamic UML diagrams of the design.

6.1 Static diagrams

6.2 Dynamic diagrams

7 Results

The results of the design work.

8 Summary

This seminar report presented the background knowledge for my gradu thesis. This seminar report will change in time slowly to my gradu thesis. After this seminar report, a decision will be made about the software process and Web application framework. Security and extensibility will be important during the decision of the server architecture.

After this point, a requirements document will be formed. During forming the requirements document, the following sub-tasks are necessary according to [Bra02]: elicitation, analysis, specification, human machine interface design and validation.

The design phase has two phases: architectural design and detailed design. UML diagrams will be drawn during the software design phase. Web application framework and the server need to be modelled in a particular way. Modelling techniques for Web applications with UML needs to be learned and used in my thesis.

Some GUI-prototypes will be implemented during the design phase, because prototyping significantly reduces requirement and design errors, especially for user interfaces [BGS84]. Database diagrams will be drawn during database design phase. On the other hand security will play an important role during design.

References

- Apa07 Apache software foundation, Apache Wicket, <http://wicket.apache.org/index.html> [19.11.2007]
- BGS84 Boehm B.W., Gray T.E., Seewaldt T.: Prototyping Versus Specifying: A Multiproject Experiment. *IEEE Trans on Software Engineering* 10, 3 (1984), 290-302
- BMU75 Boehm B.W., McClean R.K., Urfrig D.B.: Some Experience with Automated Aids to the Design of Large-Scale Reliable Software, *IEEE Trans on Software Engineering* 1, 1 (1975), 125-133
- Bra02 Bray I.K.: An Introduction to Requirements Engineering, *Pearson Education Limited*
- Bus96 Buschmann F., Pattern-oriented software architecture: a system of patterns, *Wiley, Chichester; New York*, 1996.
- Bäk07 Bächle M., Kirchberg P., Ruby on Rails, University of Cooperative Education, Ravensburg, Germany, *IEEE Software*
- ChL02 Chung S., Lee Y., Modeling Web Applications Using Java And XML Related Technologies, *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), IEEE 2002*
- HaH00 Hassan A.E., Holt R.C: A Reference Architecture for Web Servers, *Software Architecture Group (SWAG) Dept. of Computer Science University of Waterloo*, 2000
- Hib05 Hibbs C., Rolling with Ruby on Rails, 2005 <http://www.onlamp.com/pub/a/onlamp/2005/01/20/rails.html> [20.1.2005]

- LCC00 Li J., Chen J., Chen P., Modelling Web application architecture with UML, *IEEE* 2000
- Pre94 Pree, W., Meta patterns - a means for capturing the essentials of reusable object-oriented design, *Proceedings of the ECOOP, Springer-Verlag, Bologna, Italy, 1994*, pp 150-162.
- Sep02 Seppänen V., Tietoverkot course slides [PDF-document]
http://www.mit.jyu.fi/wikstrom/opetus/itk115/luennot/itk115-14_17_4slides.pdf [18.11.2007]
- ShG96 M. Shaw and D. Garlan. Software Architecture: Perspectives on an Emerging Discipline. *Prentice Hall Press*,
- ShH06 Shan C.T., Hua W.W., Taxonomy of Java Web Application Frameworks, *IEEE International Conference on e-Business Engineering (ICEBE'06)*
- Som04 Sommerville I., Software Engineering, *Addison Wesley* 2004
- Wik07a Wikimedia foundation, Web application, 2007
http://en.wikipedia.org/wiki/Web_application [4.11.2007]
- Wik07b Wikimedia foundation, Web application framework, 2007
http://en.wikipedia.org/wiki/Web_application_framework [9.11.2007]
- Wik07c Wikimedia foundation, Apache Struts, 2007
<http://en.wikipedia.org/wiki/Struts> [8.11.2007]
- Wik07d Wikimedia foundation, DotNetNuke, 2007
<http://en.wikipedia.org/wiki/Dotnetnuke> [6.11.2007]

- Wik07e Wikimedia foundation, Ruby on Rails, 2007
http://en.wikipedia.org/wiki/Ruby_on_rails [5.11.2007]
- Wik07f Wikimedia foundation, Spring framework, 2007
http://en.wikipedia.org/wiki/Spring_framework [18.10.2007]
- Wik07g Wikimedia foundation, Tapestry, 2007
<http://en.wikipedia.org/wiki/Tapestry> [26.9.07]
- Wik07h Wikimedia foundation, Zope, 2007
<http://en.wikipedia.org/wiki/Zope> [02.11.07]