

An Architecture for Web Applications

Essay in DIF 8914 Distributed Information Systems

Sven Ziemer

November 28, 2002

Abstract

Today's Web Applications are complex distributed applications, that deploy their functionality to the World Wide Web. In many cases Web Applications have been developed by using an ad hoc and unsystematic development process. As a consequence these applications have a high probability of failure during operation. In order to develop high quality Web Applications both the development process and the architecture used must take into account the features and operational environment of Web Applications.

1 Introduction

There has been an enormous success of the World Wide Web. Today many applications are developed for the web, in such different areas as banking and finance, e-commerce, education, government and entertainment. Legacy information and database systems are being migrated to Web environments, in order to deploy their functionality on the Web. Electronic commerce through the Internet is rapidly growing, cutting across national boundaries. Many people are effected by the Web.

What development approach is used for Web applications. According to [7] the development process for Web applications has been ad hoc and has lacked a rigour and systematic approach. At the same time the complexity and sophistication of web applications grows. The development cycles for Web Applications are much shorter than for more traditional software systems because for time-to-market requirements. As a consequence Web Applications have a higher probability of failure during operation.

In this essay I will describe how an adapted development for Web Applications and an architecture for Web applications can support the development of quality Web Applications. In section 2 Web Applications are described. Then the emerging field of Web Engineering will be presented (section 3) and finally an architecture for Web Applications will be provided (section 4).

2 Web Applications

The original purpose of the World Wide Web was merely presenting information. Today modern Web applications have grown into complex distributed applications. Web

applications are applications that use the Internet's infrastructure to deliver their functionality. Web Applications do not use the more traditional client/server technologies, but are using web technologies such as web browsers and web servers. Recent reports indicate that web applications represent more than thirty percent of software applications across all industry sectors [3].

Web applications are preferred over traditional applications for two reasons:

1. *Web applications are more accessible.* The HTTP protocol used in web applications is a standard protocol that can travel across corporate firewalls. The only client software a user needs is a web browser. Also, Web Applications are available on many platforms. Web browser are packaged with most operating systems these days.
2. *Web applications have a lower maintenance and deployment costs.* Since Web applications are running in web browser, they do not depend on installing client software on each user's computer. Web applications can be maintained by modifying code that resides on a server. This reduces the time and the cost of upgrade and deployment of web applications compared to traditional client/server applications.

Web applications are not limited to one type of application. They can range from simple static web pages to sophisticated applications. Different categories of web applications are grouped together according to their data and control complexity [3] as shown in figure 1:

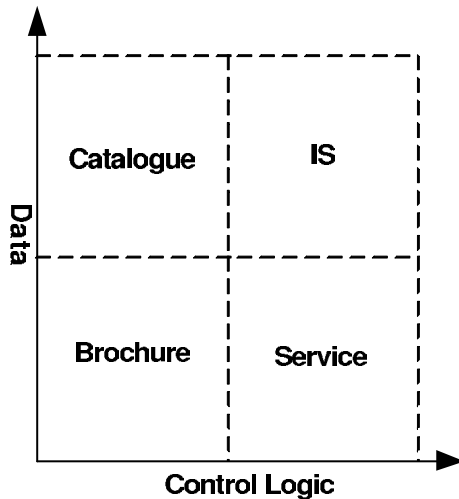


Figure 1: Taxonomy of Web Applications

1. *Brochure Web Applications:* These are the first generation of Web Applications. They are composed of static web pages and tend not to have much programming

logic in them. When developing them the focus is on content development and the layout of graphics and text.

Examples are personal web pages which contains their resume and personal information or web pages about a company's product.

2. *Service oriented applications:* These sites are offering a service to web users. Service oriented applications contains the programming logic needed to implement the service. The layout of the data is often a secondary concern. During maintenance the developers need a good understanding of the control logic.

Examples are webmail services or online word-processing systems.

3. *Data intensive applications:* These are sites that provide an interface to browse and query large amount of data. The main emphasis in these applications is on the data, with minimal amount of programming logic involved. During maintenance the developers need a good understanding of the data flow.

An example of this application type are online library catalogues.

4. *Information system applications:* These applications combine the Service Oriented Applications and the Data Intensive Applications. Developers of Information system applications are concerned with the data flow (for browsing and retrieving data) and control flow (for the different phases involved in the manipulation of the data).

Electronical book stores or online banking are example of this application type.

Developers need a good understanding of the data and control flow in their applications as needed in traditional applications. In addition, Web Applications have more dependencies and interesting relations such as the navigation links between the different pages of Web Application.

3 Web Engineering

Today's web application are offering many services that are attractive for the users. The taxonomy of web applications indicates that web applications are growing in complexity and size. This makes developing new Web Applications and maintaining existing Web Applications a major task.

Many people are using web applications as internet banking, online stores or online flight reservation systems. A good web application can a competitive advantage for a company. Many companies are depending on their web applications to stay in business. Some companies came into business through the possibilities of the World Wide Web (such as Amazon.com). Also the user's are depending on the web applications they use and the services these are offering.

Despite the importance that many Web Applications has gained for both companies and their customers, the development process of Web Applications remains rather ad-hoc [2, 7]. This means that there is no formal planning, testing and quality control and assurance. In addition, the development cycle of Web Applications is shorter than

for traditional applications because of time-to-market requirements. This makes it not easier to apply a systematic and disciplined development process.

Without a disciplined development process for Web-based applications we may face serious problems in successful development and maintenance of these applications. These applications have a high probability of failure during operation. This can cause a Web crisis [7]. This possible future Web crisis is a crisis of trust into the new web technology. To avoid this crisis there is a need for disciplined approaches and new methods for development, deployment and evolution of Web Applications. These approaches should take into account among others

- the unique features of the Web,
- the operational environment of Web Applications, and
- the diverse type (and skills and knowledge) of the people involved in developing Web Applications.

An approach to develop Web applications must also support short development cycles, which will cause a Web Application to be built in small steps.

The result of this effort should be a development process for web development. That is why there is a new emerging field of Web Engineering. A definition of Web Engineering is given by [7].

Web Engineering is the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based systems and applications.

The continuous effort to update and refine Web Applications gives rise for the term Web Gardening. Web Application development can be considered “continuous, with fine grained evolution, without specific releases as with software”[7].

4 Architecture

An important thing in a development process is to elaborate a stable and sound architecture for the system to be developed. The Rational Unified Process is an architecture centric development process. One of the evaluation criteria for the second of four major milestones is the establishment of an Architecture baseline [4]:

- Does the executable architectural baseline meet not only the requirements formally captured so far, but also the needs felt by the stakeholders as they view a working baseline?
- Does the architectural baseline appear to be robust enough to withstand the construction phase and the addition of features that later releases may require?

The point here is to show how important a sound architecture is for a software system, and that elaborating one is a demanding task. Especially the requirement that an architecture has to be robust enough to withstand the addition of future features is difficult to meet with an evolutionary development process for Web Applications. The only thing a developer of Web Applications will know for sure is that the application will change and evolve. An architecture for a Web Application must take this into consideration.

There has been some work on Web Architecture (see i.e. [1, 2]). How can an architecture for Web Applications support the development of quality Web Applications? In my opinion Web Developers should concentrate on some architectural styles that have demonstrated their stability and soundness in practice. And, given the fact that the development cycle of Web Applications is very short, web developers should start with a kind of reference architecture and carefully adopt this to their particular needs and requirements.

4.1 Software Architecture

A software architecture determines how system components are identified; the interaction between them and the interface protocols for communication [1]. A more formal definition of software architecture is given by [6]:

”The software architecture of a program or computing system is the structure of the system, which comprises software components, the externally visible properties of those components, and the relationship among them”.

This definition has two implications for distributed systems like web systems:

1. It regards the interrelationships between the components, their functional roles and the patterns of communication between them.
2. It regards the placement of the components across a network of computers, seeking to define useful patterns for the distribution of data and workload.

4.1.1 Software layers

Software is structured as layers in a single computer or as services offered and requested between processes located in the same or different computers. This process- and service-oriented view can be expressed in terms of service layers.

4.1.2 System architectures

The division of responsibilities between system components and the placement of the components on computers in the network is perhaps the most evident aspect of distributed system design. Two main types of architectural models are the *client-server model* (figure 2) and *Services provided by multiple servers* (figure 3).

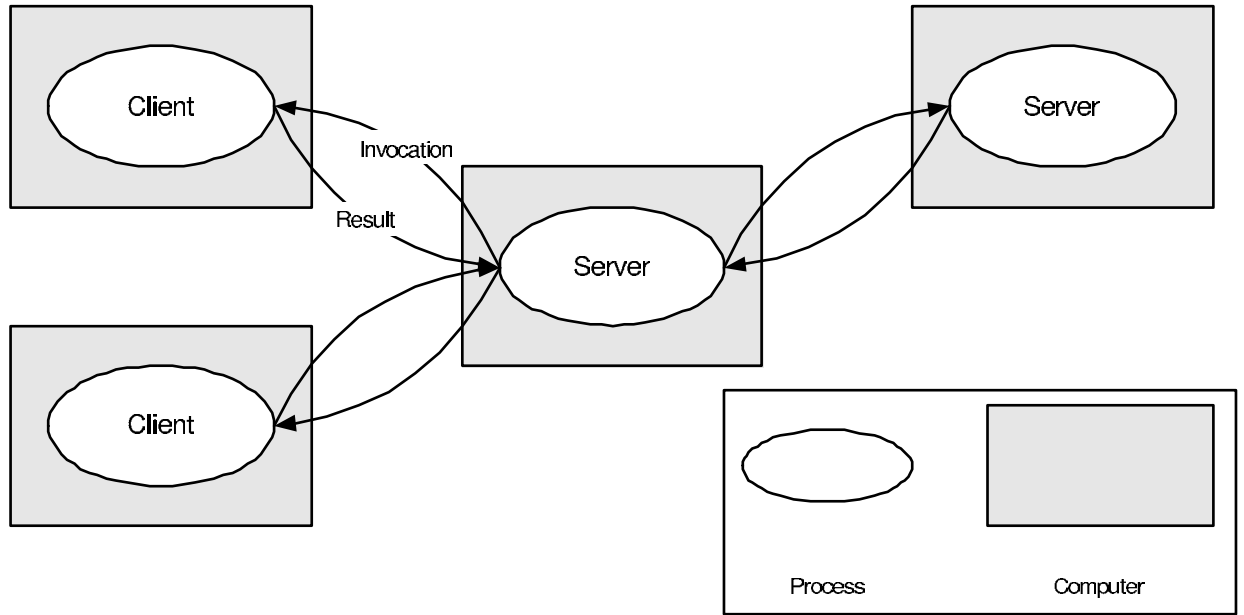


Figure 2: Client-Server model

The **Client-server model** is historically the most important architectural model and remains the most widely employed. A client process interacts with individual server processes in separate host computers. Servers can be clients of other servers.

Services provided by multiple servers A client interacts with a server that provides a service to the client. This server interacts with other servers to provide this service, because the servers may partition the set of objects on which the service is based.

4.2 Multiple Architecture Views

A software architecture addresses many concerns and is used by many stakeholders. To avoid cluttering the architecture diagram with too many details, it has been proposed to use different architecture views, that address the specific concerns of that view.

Philippe Kruchten [5] proposes modelling architecture using four different views and one use-cases view to illustrate and validate the other views. In his *4+1 View Model of Architecture* he defines the following views:

- **The Logical View** This view of the architecture addresses the functional requirements of the system. The model used in this view is the design model, and identifies design packages, subsystems and classes.

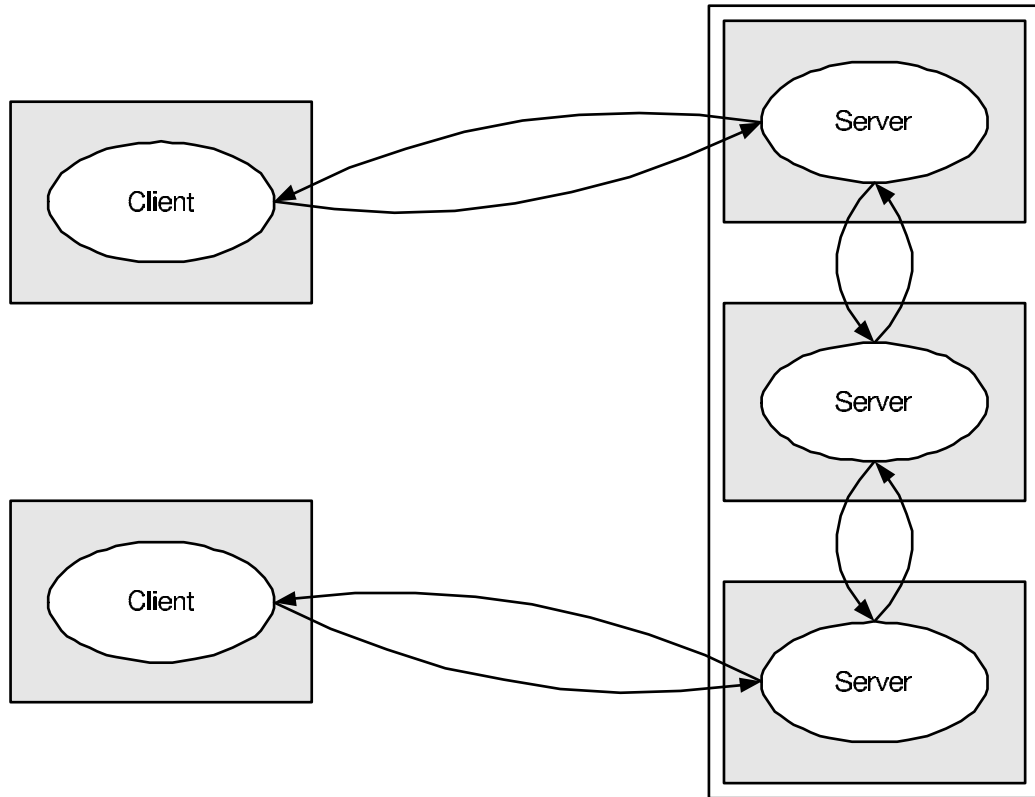


Figure 3: A service provided by multiple servers

- **The Development View** This view describes the organization of static software modules in the development environment in terms of packaging.
- **The Process View** This view addresses the concurrent aspects of the system at run-time – tasks, threads, or processes as well as their interactions.
- **The Physical View** This view shows how the various executables and other running components are mapped to the underlying platforms or computing nodes.
- **The Use-Case View** This view contains a few key use cases, that are used to drive the discovery and design of the architecture in the early phases. In the later phases they will be used to validate the different views of the architecture.

4.3 Architecture of the Web environment

Web Applications exists in the special environment of the World Wide Web, and have take into account the special feature of it. The developer have to understand this environment.

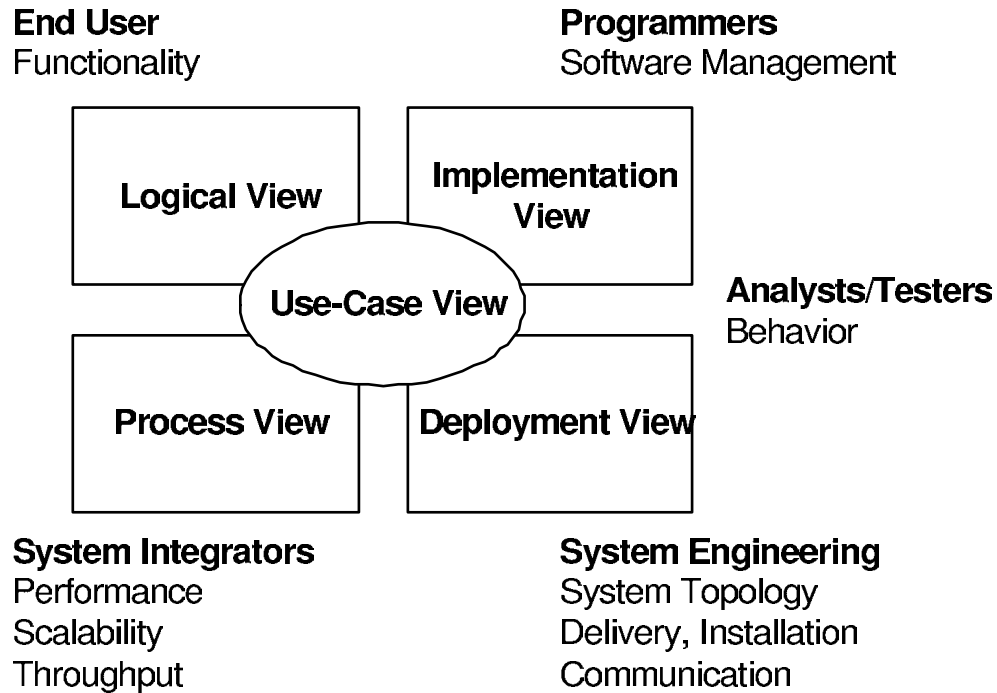


Figure 4: The 4+1 View Model of Architecture

The *goals of the WWW project* was to establish a shared information space through which people and machines could communicate [1]. People intending to use this system were located around the world. The information to be shared was ranging over many different types, such as text documents, pictures, etc. The challenge was to build a system that would provide a universally consistent interface to this structured information. The Web's information sources would be distributed across the global internet. Therefore the architecture needed to minimize the interactions with the network.

The architecture had some main concerns:

- How to deal with the latency that comes from the network interactions.
- Scalability, since some resources would be particularly newsworthy and lead to sudden spikes in access attempts.
- How to introduce a new set of functionality to an architecture that is already widely deployed.

This motivated the development of the Representative State Transfer (REST) architecture [1]. This architecture is an abstraction of the architectural elements within a distributed hyper media system. It has three classes of architectural elements.

Data elements In REST all data is moved to the location where it will be used. The components communicate by transferring the representation of the data in a

format matching one of an evolving set of standard data types, selected dynamically based on the capabilities or desires of the recipient and the nature of the data.

Processing elements The REST components are typed by their roles in an overall application. *User agents* uses a client to initiate a request and becomes the recipient of the response. An *origin server* is the source for representations of its resources. Other types of components are gateway's and proxies.

Connecting elements REST uses various connector types to encapsulate the activities of resources and transferring resource representations. All interactions are stateless. Primary connector types are client and server. Other types of connectors are cache, resolver and tunnel.

4.4 Web Application Architecture Views

A Web Application Architecture view can build upon the four views proposed by Kruchten: Logical, Process, Physical, and Development view. Each view captures specific decisions and all views must be examined together to gain a good understanding of the whole applications. To account additional requirements such as security and requirement, additional views can be added to the Web Application Architecture View.

- **The Logical View** The Logical view provides a high level abstraction of the system based on the domain of the problem. The application is represented by different components and the interaction between them. At the architectural level, Web Applications can be organized as 2-tiers or as 3 tiers (as shown in Figure).

The *Presentation Logic* tier shows the interaction between the components responsible for the generation of the User Interface. Components in this tier only interact with components in the *Business Logic* tier. The components in the *Business Logic* tier contains all the knowledge required to modify the data components that are contained in the *Database* tier. The *Database* tier contains all the data components that are used to provide persistent storage for the application data.

The 3-tiered architecture provides a good separation of concerns. In a 2-tiered architecture both business logic and data access code are mixed. The advantage of the 3-tiered architecture is the encapsulation of concerns. The database system used for the implementation can easily be changed without affecting the Business Logic (as would be the case for a 2-tier architecture).

- **The Development View** The Development view focuses on the mapping of the Logical view conceptual components to the actual implementation artifacts. It presents that actual software module organization in the development environment.

The Development View for Web Applications must highlight additional details such as:

- The link structure of the application pages

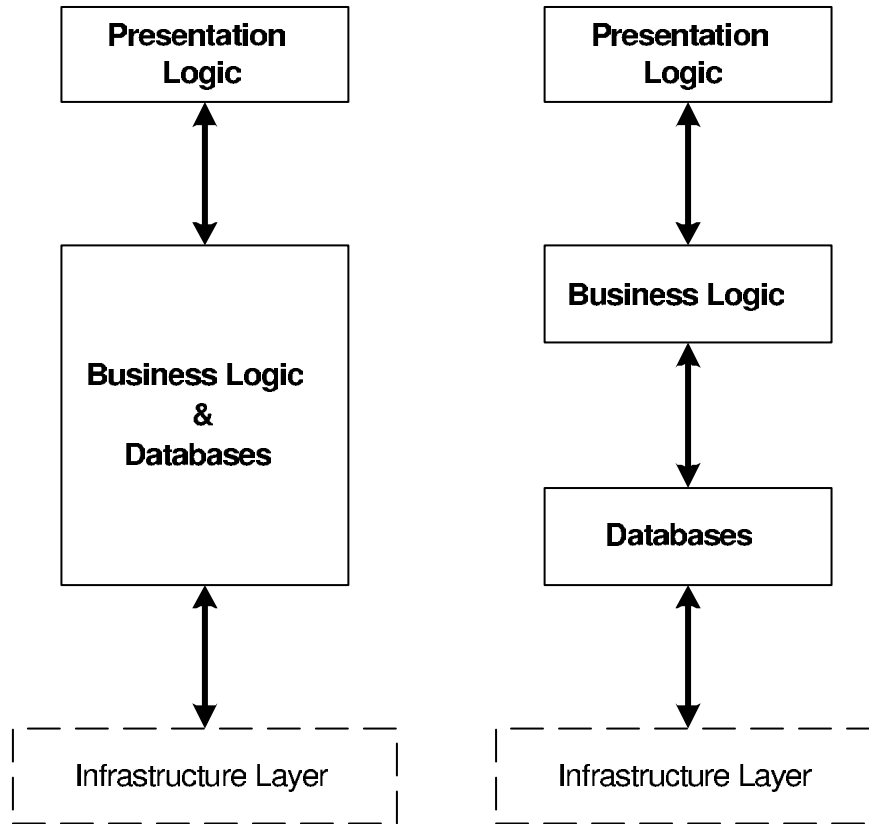


Figure 5: High Level Logical View of a Web Application

- User's session management techniques
- Application page generation technology

Many web pages are linked together to form a Web Application. To avoid death links all links have to be checked regularly.

Web Applications use the HTTP protocol as their communication medium. Since the HTTP protocol is a stateless protocol, the Web Application has to use techniques such as Cookies or hidden fields in a web page to store the state for a particular session.

Application pages are a mixture of HTML tags and control code. The control code is used to personalize web pages and the HTML tags are used to format the output of the page. When an application page is requested, a web server is preprocessing all data from various resources and generates the final HTML page. The developer has two options for developing application pages: First, he can use *Code based* application pages. With this technique HTML pages are generated fully by executable programs. Some examples of this technology are:

CGI and Java Servlet. Second, the developer can use *templated-based* application pages. These are written with the HTML Language and extended with tags to embed control code. Examples of this technology are PHP and Java Server Pages (JSP).

- **The Physical View** The Physical view presents the mappings of the components in the Development view to the components in the environment. Web applications have a rich environment, which contains the following components (see also Figur):

- Web browsers
- Web servers
- Application servers
- Databases
- Distributes objects (i.e. Enterprise Java Beans)

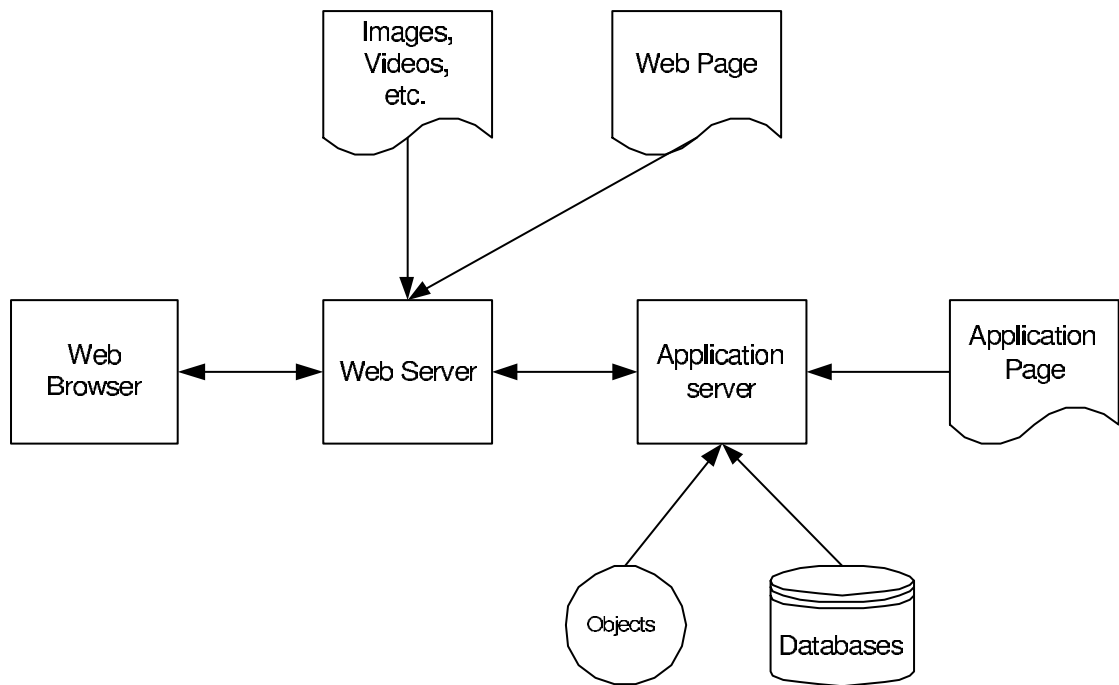


Figure 6: Physical View of a Web Application

The user of a Web application uses the web browser as the interface to get access to the Web applications functionality. The browser transmits the user's action to the web server, sending the requests using the HTTP protocol. The web server determines if the request can be fulfilled directly. Otherwise the applications

server must be invoked. As can be seen from figur, if the user wants to retrieve some data from the database, the application server must be invoked. Finally, the web server with the possible returns from the application server generates the HTML pages that tis returned to the user.

- **The Process View** The Process view presents the concurrency and distribution of process in the application.

5 Conclusions

This essay described the field of Web Applications and presented the needs for an own engineering approach for such systems. This approach, called Web Engineering, should have an associated architecture, that takes into account the special features and needs for the development of Web Applications. A architectural framework for Web Appicaitons using architecural views was presented.

References

- [1] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. In *ICSE 2000 Limerick Ireland, 2000*.
- [2] Guntram Graef and martin Gaedke. An evolution-oriented architecture for web applications. In *Second Nordic Workshop on Software Architectureb(NOSA 1999)*, 1999.
- [3] Ahmed E. Hassan. Architecture recovery of web applications. Master's thesis, University of Waterloo, 2001.
- [4] Grady Booch Ivar Jacobsen and James Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999.
- [5] Philippe Kruchten. The 4+1 view model of architecture. *IEEE Software* 12(6), 12(6), 1995.
- [6] Paul Clements Len Bass and Rick Kazman. *Software Architecture in Practice*. Addison Wesley, 1998.
- [7] S. Hansen S. Murugesan, Y. Deshpande and A. Ginige. Web engineering: A new discipline for development of web-based systems. In *Proceedings of the First ICSE Workshop en Web Engineering, 1999*.