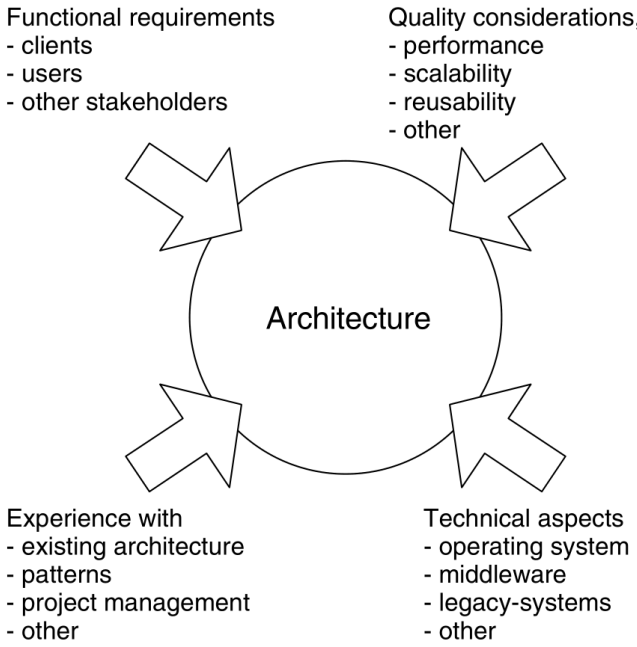
Part II. Architectures for the Web

Architecture is defined as the structure of components, their relationships, and the principles and guidelines governing their design evolution over time. According to (Bass et al. 1998), the architecture of a software system consists of its structures, the decomposition into components, and their interfaces and relationships.

When we create architecture we try to break the functional requirements and quality requirements down into software components and their relationships through interfaces using an iterative approach. Depending on the point of view, we can emphasize and itemize different architectural aspects. Structuring software systems and breaking them down into different perspectives allows us to better manage the complexity of software systems, and the systems become easier to understand. In addition, the abstraction of system aspects facilitates the outline of possible and important architectural issues.

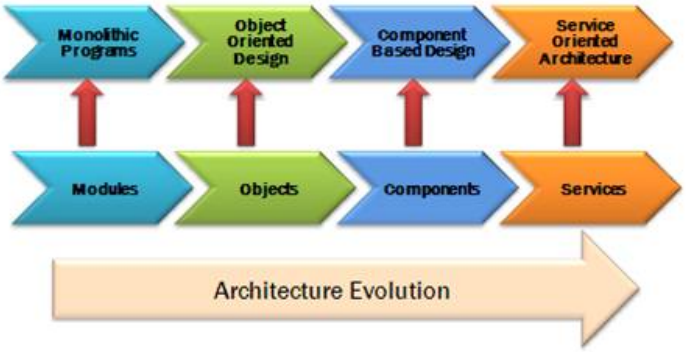
Considering the above properties of architecture we can easily see that architectural decisions are of enormous importance for the development of Web applications. The requirements of software and thus its architecture are subject to change. Technical and organizational constraints change during and after the development of an application. This is the reason why software systems are often referred to as "moving targets". Due to the changes in capability and performance objectives, interests of different stakeholders, technology improvements, and unanticipated situations, existing architectures may need to evolve to meet new capability requirements and constraints. The following figure shows the different factors and constraints influencing the development of an architecture according to (Jacobson et al. 1999).



Factors influencing the development of an architecture

The evolution of distributed architecture can be considered a sophisticated extension of client/server architecture because the Internet and World Wide Web caused a shift in the nature and ways of business transactions in place today. With this evolution, especially when it concerns to the Web, came a large quantity of services and tools. As these services and tools grew up in uses and complexity they came to be

the subject of software engineering, a topic that was well developed by the time this applications became used. The evolution of architectures could be imagined with the following figure:



Evolution of Architectures

A number of architectures for specific requirements in several application domains have been developed in the past few years. The most widely used aspect is the layered view. Layering means that software systems are structured in several tiers to implement the principle of "separation of concerns" within a software system. Many frameworks in the field of distributed systems and Web applications are primarily structured by the layering aspect, which will be discussed in the next chapter.

The increasing distribution of software systems has led to the development of architectures and infrastructures addressing the distribution of data and messages. We could found several proposals to achieve an efficient way in this distributed nature. The most well-known architectures are:

- Distributed Object Middleware (DOM): This type of infrastructure allows to access remote objects transparently. It is based on the Remote Procedure Call (RPC) mechanism. Some DOM systems also enable objects on different platforms to interact (e.g., CORBA). Other examples of this type of system include Microsoft's DCOM (Distributed Component Object Model), or EJB (Enterprise Java Beans) by Sun Microsystems.
- Message Oriented Middleware (MOM): MOM systems offer functionalities for asynchronous transmission of messages. Asynchronous communication are sent to the receiver regardless of its status, e.g., the receiver may not be available when the message is sent. MOM ensures that messages are delivered nevertheless. Examples of MOM systems include Sun's JMS (Java Messaging Service) and Microsoft's MSMQ (Microsoft Message Queue).
- Peer to Peer (P2P): P2P stands for direct communication between two devices – the peers – in a system without using a server, i.e.,they communicate over a point-to-point connection. The peers are basically equal. P2P systems describe how the devices in such a network communicate and how they can "discover" each other.
- Service Oriented Middleware (SOM): SOM enhances DOM systems by the concept of services. A service in this context is a number of objects and their behavior. These objects use a defined interface to make a service available for other systems/services. SOM defines communication protocols between services, and provides for location- and migration-transparent access to services, thus supporting a simple integration of services beyond platform boundaries. One example of a SOM is Sun's Jini system (http://www.sun.com/software/jini/). Architectures emerging within the field of Web services also belong to this category.

These architectures are applicable to distributed systems in general, which means that they are not limited to Web applications. Similarly, integration architectures address integration aspects on the content level and the application logic level and are commonly summarized under the term Enterprise Application Integration (EAI) architectures. This category also includes architectures which integrate existing applications as a whole. Alternatives to EAI are Web services which support the integration of services, i.e., application logics

and contents. On the presentation level, a set of different systems is typically integrated by using portal architectures.

Table of Contents