# Designing Mobile Application Servers

## A Prototype Approach Applied to the Construction Industry

***Michel Nilsson***
***Björn Ryding***

Master Thesis 20p 2002 IA7400
Tutor: Ola Henfridsson, PhD

Department of Informatics
School of Economics and Commercial Law
GOTEBORG UNIVERSITY

# Abstract

The use of mobile computing within in the construction industry is an emerging field within informatics and software engineering research. This thesis presents a system design of an application server, which is thought to act as a gateway between existing information systems and mobile devices used within in a construction industry context and especially on a construction site. The purpose of the project was to investigate how a mobile application server should be designed. The system design focuses on aspects like integration with existing information systems, synchronization matters, vendor independence, platform independence and generic support for existing and future mobile clients. Today, there are a number of existing and emerging technologies to choose from when designing new IT artefacts. In this project the Java 2 Enterprise Edition framework and various XML frameworks, have been applied. The mobile applications server is based on an application server following the J2EE specification. Besides the above mentioned technologies the Aptus telematics platform from Pilotfish, which incorporates telematic functionality in the system design in the form of machine connectivity have been used. The results are presented in the form of a general design, which is vendor independent, with regard to software components and third party frameworks, and a specified design containing various products found relevant for a future implementation. The specified design is platform independent since all of the components used are pure java software. Since the work has been design oriented the main research approach has been to apply evolutionary prototyping methods, design patterns and software engineering best practice. Besides the general and the specified design the results also includes some prototype applications implemented on a Compaq iPAQ running SavaJe OS, which provides a fast J2SE compliant runtime environment.

## Acknowledgements

We would like to thank our tutor Ola Henfridsson, who has supported and guided us on the journey while completing this thesis. We would also like to thank the Viktoria Institute in Göteborg where the major part of the work with this master project was conducted. The Viktoria Institute kindly supplied us with all the computer equipment used during the work. Finally we would like to thank Pilotfish for supplying us with their telematics platform.

Michel Nilsson & Björn Ryding

*Göteborg, October 2002*

# 1. Introduction

This master thesis project was conducted during June – September 2002. The work addresses the need for an application server in order to facilitate mobile computing in the construction industry and essentially the use of mobile devices such as Personal Digital Assistants and Mobile Phones. The introduction chapter introduces the problem and specifies the research question.

## *1.1 IT in Construction*

The construction industry is interesting because of several reasons. Construction projects are complex and a lot of different organizations and people are involved in construction projects. Contractors, subcontractors, architects and customers have to collaborate during a set time period in order to complete a project. There is a considerable need for communication between humans, machines and existing information systems. Unlike manufacturers who can fine-tune processes and create permanent production systems, construction firms are relegated to existence in a wilderness of changing project infrastructures (Finch, 2000).

For most industrial countries civil engineering is of major importance to their economies. Compared with other production industries, like mechanical or electronic engineering, civil engineering can be characterized as producing mainly one-of-a-kind objects (Partsch, 1998). During the production of one of these one-of-a-kind objects a large number of participants are involved over a time-period ranging from a couple of months to several years. Cooperation along all participants in a construction project is essential, in order to achieve cost efficient production and on time completion.

Since the industrial revolution started in the 18th century, machines have been used to automate and to improve material handling tasks (Björk, 1999). The construction industry shows an increasing trend to use machines to automate both material and information handling processes. During the later part of the 20th century machines have increasingly been used to aid and improve the information-processing task. According to Björk (1999), early uses were in particular computer applications for engineering analysis. Today, IT use in the form of CAD, word processing software, copying machines, faxes, mobile phones, computer networks etc. have a huge impact on all aspects of the information process. The technological development in the last few decades have elevated the role that information plays in the operation and management of companies, and is causing a rethink in the way companies in general treat information and the systems handling this information (Harris & McCaffer, 2001).

Software vendors today are focusing on integration, synchronization and the support of mobility. New and different types of software are emerging, especially in the area

of e-commerce and project collaboration. How do all these different applications fit and work with each other and with the software in place within any one company? There is a lot of talk about end-to-end solutions, but no one is offering a set of applications that addresses the spectrum of needs for a construction firm in a comprehensive, integrated manner (Koenig, 2001).

## *1.2 Mobile Computing*

Mobility or mobile computing describes where a device like a mobile phone, personal digital assistant (PDA) or smart phone can connect to a network (e.g. LAN, internet) or a computer to synchronise applications and information. Mobile computing also refers to computers contained in commonplace objects such as cars and appliances and implies that people are unaware of there presence. Devices will communicate with each other without any interaction required by the user.

PDAs emerged in the nineties and constitute a large user base within the mobility market, with many applications having been developed for the platform. Most models have a colour display that is backlit. They typically have between 8 and 64 MB of RAM memory, and can be expanded as required. The most recent models have built-in modems for wireless Internet access. User interaction is via a stylus and built-in buttons for navigating and managing applications. Additional peripherals can be added to expand the PDAs capabilities including cameras, global positioning systems, keyboards, barcode scanners and printers.

Several software companies offer applications for the construction industry, which are developed for mobile platforms e.g. Autodesk's Onsite View, which offers viewing, mark-up design changes, on-site project document quires using digital measurement tools, and synchronisation. Geographic Information Systems (GIS) are already available for some PDAs as well. These types of applications work fine as standalone applications and they are certainly a welcome improvement but to truly utilise the possibilities with mobile computing a transition to structured information would be necessary.

## *1.3 Problem*

Information created and stored by specific software applications cannot, in most cases, be brought into other software applications. The construction industry uses a number of different types and makes of software applications. Applications include planning and scheduling, cost modelling, computer aided design, engineering design and many others. Considerable time and effort are spent entering information into these systems and the outputs that they produce are often only capable of being reused within the software environment they were created in. Information is created by a number of different participants and organizations in a construction project, each using there own proprietary solutions and operating systems. Information is

created at many different stages in the construction lifecycle. Without the tools to transfer information between packages used at different stages, much of the information is either lost or recreated at great expense. For project data that is available online there is still no effective way of searching documents in a sensible way. The potential of the computer environment to screen and filter information is lost because no consistent method is used to structure information in a way that can be understood by machines. There have been many false dawns in the construction industry with respect to IT (Finch, 2000). The arrival of new standards and software solutions has promised seamless integration between software applications and companies, but the reality has been very different.

Looking at the mobile perspective in construction IT there have been some minor, typically isolated IT solutions which did have a tremendous, albeit not mainstream impact on the construction industry e.g. telefax machines and mobile phones. In these cases the actual automation level of the process did not increase at all, but distance was no longer a problem (Magdic, 2002).

We believe that a lot of available technologies such as mobile devices and wireless network components have matured significantly in the last couple of years. In order to facilitate mobile computing for the construction industry the problem is rather on the actual server side.

## *1.4 Research Question*

How should a software platform, that is to act as a bridge between existing information systems and mobile devices used in a construction industry context, be designed? The main question is broken down into the following subquestions.

1. Which information exchange standards are to be supported?

2. How can platform and vendor independence be achieved?

3. How should the system handle multithreading, synchronization, transactions, and resource allocation management?

4. How should the system support existing and coming mobile clients?

5. How should the system ensure scalability and maintainability?

Our focus is on the system design and the technologies needed to implement such a design by applying best practices design strategies. We are not concerned with questions like: how does mobile computing work on site, and what organizational changes are needed when implementing mobile IT support in the construction industry? These types of questions are often addressed in the informatics research domain but they are not within the scope of this project.

## *1.5 Expected Result*

We expect to present an architectural design describing a software system that supports the use of mobile computing in a construction site environment, where we have studied the possibilities with existing and emerging technologies. Further we expect to present results concerning the possible use of these technologies in future real world system implementations.

## *1.6 Choice of Research Area*

The construction industry is very much characterized as one that produces very large amounts of data. Managing, handling and insuring accuracy and availability of the information produced from the vast data volume is crucial for the successful completion of any construction project. Inaccuracies and missing information can lead to project delays, uneconomical decisions, or even failure of a project. The collection and the distribution of onsite information is a critical concern to all the participants of a construction project. One of the major problems in using information technology in civil engineering is the fact that production activity is dispersed and site locations frequently change. Compared with other industries, this has proved a great disadvantage in giving construction sites adequate IT support.

Most construction IT solutions, like integrated building models, including complex process and product models, require highly organized and standardized project environments which are not found in real-life construction projects. Participants in a construction project are very often at very different levels of organization and IT use (Magdic et al, 2002). They are usually forced to use mutual digital communication at quite a low level.

The possibilities with wireless infrastructure, system integration and synchronization in the field of construction IT is a very interesting notion to study and the fact that very little research has been conducted within this context was one of the major reasons for us to approach this area. Further we have been inspired by the technology push from the computer industry and essentially software and technologies based on the Java programming language like the Apache XML project.

# 2 Method

This chapter describes the research method found applicable to our problem. The method applied is very much inspired of software engineering methods and specially methods addressing software design.

## *2.1 Course of Action*

Our problem concerns mobile computing in constructing and essentially the design of a system to support the operations management on a construction site. The study is design-oriented. Our main objective is to present a conceptual system design and from the conceptual design via prototyping present a specified design containing of-the-shelf-software products and third party frameworks. Our goal is not to produce a working prototype, but merely to apply prototyping as a method to generate the design specification. With our system design we aim to address specific design issues concerning the development of mobile systems in a construction context and essentially answer our research question.

Informatics can be described as "applied computer science" (Holmquist, 2000) and it has its roots in the discipline of information systems research. Informatics has its core in information technology (IT) use and the design of new IT. Informatics is a design-oriented discipline (Dahlbom, 1996). The design focus in informatics follows from the notion that it is an artificial science i.e. it is concerned with objects created by man, as opposed to the natural sciences, which study things in nature. In the process of designing new IT-artefacts, informatics is to be viewed as systematic, institutionalised form of such a design activity (Dahlbom, 1999).

Our problem concerns computer technologies and the possibilities for integration, development and design of systems using these technologies. Approaching the problem from this perspective, we argue that the core of the problem is very much located in the technical domain of software engineering, traditionally often applying a positivistic research approach (Patel, & Davidsson, 1994). A lot of research conducted within the informatics domain applies a holistic view of the system and the research question is often concerned with how the design will affect the actual future use of the system. When performing this type of studies, certainly interpreting methodologies are applicable (Wiederheim-Paul & Eriksson, 1991). However, since our focus is on the system design itself, and not on the implications of such a system, this approach is clearly not suitable in this case.

## *2.2 Research Model*

Mobile computing in construction is a rather new concept and we have not been able to find an implemented system within an organisation or a company, to conduct a case study. Instead of doing a case study we will perform observations during the development of our system design in order to evaluate the technologies and the methods applied. The objective with the implementation of a prototype system is to evaluate the validity of the chosen design and to improve the design in an iterative, evolutionary process. The design process and the development process will be conducted concurrently and the actual evaluation will be performed in concurrency with these processes. This differs from what may be the conventional case when evaluating a design concerning a computer system. The concurrent evaluation approach is proposed in lightweight software engineering methodologies such as extreme programming.

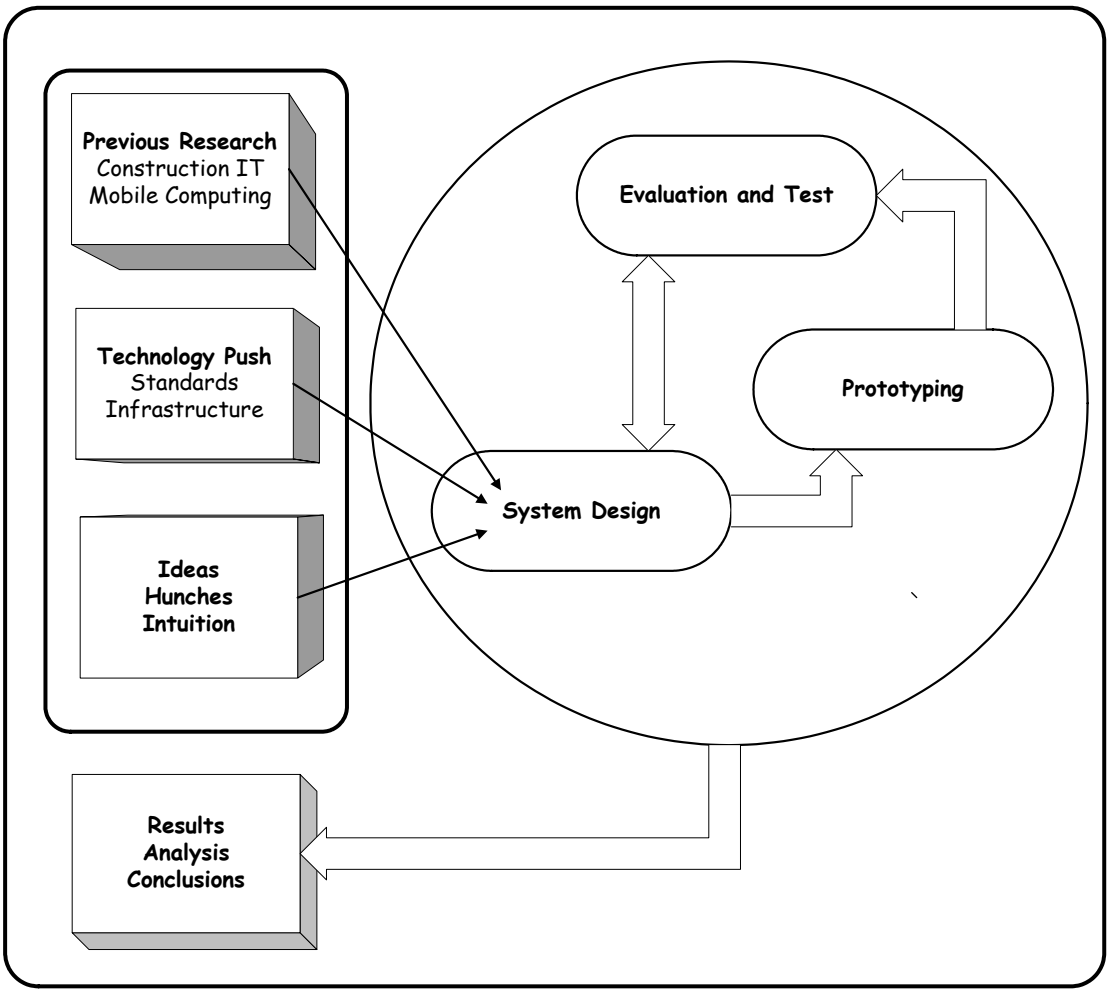



**Figure 2.1 Research Model**

## 2.3 Literature Study

A literature study is usually the start of any research project. With the literature study our objective is to identify relevant parameters to study and to constrain our problem definition. Further we want to confirm that the core of our problem and our research questions are "new" and that the relevant questions, have not already been answered by other researchers. Sekaran (1992) stresses the importance of conducting a thorough literature study in order to get the most relevant and updated picture of the conditions in the problem domain. The work of extracting information can continue during the other phases of a research project.

During the literature study will mostly focus on recent research in construction IT and papers concerning mobile computing within construction IT. Our main objective is to find theories, information models, design frameworks, design implications and other relevant results that we can use and develop in our system design. Besides construction IT we will also focus on general software engineering issues, including design patterns, system architectures, implementation technologies and industry standards for information interchange like different dialects of XML developed for the construction industry.

## 2.4 System Design

Using the findings from the literature study and combining it with our on ides our goal is to produce a system design. We will in general use an evolutionary approach in our design process and in the implementation process. Instead of having separate specification, development and validation activities these are carried out concurrently. Using this method we will get rapid feedback across these activities (Sommerville, 2001). The design approach will be component based incorporating several of-the-shelf software and hardware products including, a XML publishing framework, a application server, a telematics platform and others.

The design specification will contain an architectural design, a process design and a component design. Sommerville means that the design process used depends on the application and the skill and intuition of the developer (Sommerville, 2001). Sommerville suggests the following activities when developing an architectural system design.

*System Structuring.* The system is structured into a number of principal sub-systems where a sub-system is an independent software unit.

*Control Modelling.* A general model of the control relationships between the parts of the system is established.

*Modular Decomposition.* Each identified sub-system is decomposed into modules. The developer decides the type of modules and the interconnectivity.

We will produce a process design in order to establish the collaboration and synchronisation matters concerning active objects, their execution and scope. The need of a process design is very system dependent (Mathiasen et al, 1999). In stand-alone administrational system the process design is not a big issue whilst on system utilizing parallelism the process design is of considerable importance.

The component design will contain descriptions of the functionality offered by the components. Designing the components we will use an extended version the three-layer solution proposed by Mathiasen et al (1999), which divides the functionality into a model component, a function component and an interface component.

## *2.5 Specified Design and Prototyping*

Sommerville (2001) specifies three different methods for software prototyping: *evolutionary prototyping, throwaway prototyping* and *incremental prototyping.* In evolutionary prototyping the system is developed and improved in an iterative process until the final result is established. In throwaway prototyping the main objective is to develop the specification and the actual prototype is not used in the final system, as opposed to the previous method, hence the name throwaway. Incremental prototyping is used when developing large system where the development is separated into different modules and sub systems.

Preece et al (1994) describes prototyping using the terms full, vertical and horizontal. Full prototyping means a system with full functionality but with lower performance. Horizontal prototyping focuses on the user interfaces but not the implementation of the functionality behind the interfaces. Vertical prototyping means full functionality within the implemented subset of the specified system. Our approach is to use evolutionary prototyping in conjunction with vertical prototyping. Hence the objective is to develop full functionality within the chosen sub set of our proposed design.

All programming will be done in java, which is the language of our choice. All of the software components that are incorporated in the design are Java based and using Java will ensure platform independence and code portability. Java is a strongly typed, object oriented programming language, which has matured considerable since its introduction in the mid 1990s. It contains a number of useful API's for software development in distributed and networked environments.

# 3 Related Work

This chapter presents related and ongoing research within the field of construction IT and mobile computing in construction. Several leading research projects are being undertaken by academic and research institutions around the world on the subject of construction IT. Much of the work derives from a broad information technology base but some work has been carried out in the area of mobile computing.

## *3.1 General Construction IT*

In his paper *Information Technology in Construction: Domain Definition* Björk (1999) discusses the scope of research on information technology applications in construction. He presents a model of the construction process constituted of two main groups of activities, namely the information activities and the material activities. This model is used as foundation for a definition of construction IT research which is thought to facilitate and re-engineer the information process component in construction. Björk (1999) means that IT includes all the technologies used to store, transfer and manipulate information taking into account devices such as copying machines, faxes and mobile phones. The scope of construction IT is compared with the scope in related areas such as design methodology, construction management and facilities management. As a comparable application domain for IT and process improvement, Björk (1999) argues that healthcare is an interesting alternative to the often used car manufacturing industry. He further discusses some of the key areas for construction IT research during the mid 90s an onward such as IT strategies, expert systems and product modelling.

Aoued el al (1999) have investigated the technological shift in viewing IT as driver for many of the construction business and operation processes, to the viewing of IT as an enabler for these processes. According to Aoued et al (1999) the technological management of IT within the construction industry has been given little attention in the past. As a result a lot of IT solutions have been developed to act as drivers in the design and construction processes. The authors suggests that IT enabling solutions will play a major role in achieving major improvements in a traditionally fragmented design and construction processes. Aoud et al (1999) presents an IT process map that illustrates how IT could operate within a process framework. This requires an agreement on the actual design and construction phases, structure and management. In such a way the process becomes the driver and IT the enabler. The authors argues that by applying there proposed framework, it is anticipated that construction firms will move away from traditional ad hoc IT investments and move towards well planed strategies. By doing so large, as well as small organisations would be able to identify opportunities for IT-investments, evaluate their existing systems, identify the rate at which new IT applications are adopted, and work out the level of impact of IT on their firms.

Koenig (2001) performs a survey of software applications targeted at construction companies. The primary focus is on business related applications such as accounting, project management, administration and collaboration. The study also touches on CAD, estimating and the scheduling aspects of project management. Koenig (2001) presents three groups of software applications used within the construction industry: Enterprise Resource Planning (ERP), Industry Specialists and PC applications. The first group, ERP systems are very large scale, integrated applications that intends to offer a total solution for all of a company's management information needs. Originally designed for Fortune 500 companies in industries such as manufacturing, distribution and financial services, they are now finding ways into the very largest construction and engineering firms. Vendors of ERP-systems include J.D. Edwards, Baan, Oracle, PeopleSoft and SAP. ERP-applications have been characterised by high cost and complexity, which can be considered a natural by product of trying to address the complete range of business functions across a wide spectrum of industries.

The second group of construction industry software applications come from vendors that fall into the category of industry specialists. These vendors mainly address the market containing large contractors, with relatively robust, construction-specific application suites. According to Koenig (2001) most of these vendors offer comprehensive function sets with relatively slight differences, sometimes oriented towards particular types of contractors. Some of these applications are able to address the more complex requirements such as multiple business units, multiple offices or locations, diverse project mix and fully integrated modules.

The third group – PC applications contains all types of applications that are specially design for small to medium sized construction companies. These applications are in most cases well scaled in terms of price, functionality, and ease of use to small to medium sized construction firms. Due to the fact that this market is much larger compared with the market for group one and two type of software, there are many more vendors serving it with PC applications. The clear market leader is Timberline (Timberline, 2002), a company that was the first player of any size to put a Windows based construction application on the market.

## *3.2 Mobile IT in Construction*

Magdic et al (2002) discusses the concept of mobile computing in construction. The first part of the paper defines the concept of mobile computing and presents the potential use for the construction industry. The second part describes a case study that was performed at a real building site were the systematic use of mobile computing was investigated. The case study showed that the efficiency of information exchange with and at a building site can be improved significantly by using current unmodified mobile computing components like PDAs, mobile phones and Internet services. Magdic (2002) means that the stressed conditions at a building site in terms of dust, strong light, rain etc. site requires special equipment and the

devices used during the project were not suitable for the environment in question. However, lately there have emerged several vendors that produce heavy-duty handheld devices.

According to Magdic (2002) one significant improvement would be the use of structured information instead of conventional documents. In this way information automation and integration would be possible. During the case study only selected documents could be interpreted using the selected software, and no data integration was possible. Non-standard data formats also made it necessary t convert some document types so that they would be usable with running software on a PDA. Magic (2002) concludes that a standard way of structuring and describing data, like XML, could solve these problems.

Ybuki el al (2002) have developed a prototype system for on-site inspection. In there research they present a new system model for supporting on-site inspection of buildings and facilities by using and combining information technologies including Radio Frequency identification Tags (RFID), voice input-output, wireless LAN, the Internet and knowledge management by using VoiceXML (VoiceXML, 2002).

In the system model a field inspector carries a PDA, a cellular phone and a digital camera. In the system the PDA has a "reader-writer" that can read and write information into the RFID tags. The RFID tag contains an integrated circuit and an antenna. These tags are attached to the various facilities and an inspector can receive and store information in the tags. The general idée is to work with local facility information stored in various tags, and at the same time have access to global information via wireless networks.

Cox et al (2002) presents work addressing construction filed data inspection using handheld devices. In there paper they describe an application for automating the collection process of field inspection data using a PDA. The application allows for recording, processing and distribution of inspection information of various tasks performed in the field. Cox et al (2002) argues that the major benefits from the automation of the field data collection process are; the reduction of paperwork, automatic generation of reports, and faster distribution of electronic data. The author's stresses the point that the implementation of new technologies is always followed by a number of challenges. These challenges ranges form usability and personal usage to organisational issues. Cox et al are involved in ongoing research investigating the use of handheld devices with wireless technologies for the transmission and distribution of construction data directly from the field.

# 4 Theoretical Background

In order to develop a design and to implement a prototype it is necessary to apply some theoretical design principles. Even simple software systems have a high inherent complexity; so engineering principals have to be used in their development. This chapter discusses different software engineering methods that are of relevance to the problem. The first section presents general ideas regarding software engineering and the following sections focuses on software engineering methods for distributed systems, information exchange and mobile computing. The intention is not to study, improve or evaluate software processes and development methodologies but to systematically apply what is commonly referred to as best practices (Alur et al, 2001).

## *4.1 Software Engineering Principles*

It is useful to distinguish two different levels at which software engineering principles apply: those concerning small programs and those concerning large software systems. The first is called programming-in-the-small. The second is called programming in the large.

Programming in the small concerns concepts that help developers create small programs, those that vary in length from lines to a few pages. Sometimes the creation of small programs requires great precision and ingenuity. It is important to be able to reason about small programs with clarity and mental precession in order to design them effectively.

In software design designers are also interested in ideas for program structuring that help make the flow of control during program execution mirror the textual structure of the program. By using good program structuring the reasoning about programs is made easier and made more effective, and programs are made easier to understand. One technique that is useful for creating structured programs is called top-down programming by stepwise refinement; it is applied by starting with an outline of a program's major features and filling in the lower levels of detail.

In programming in the large system developers are concerned with principals that organise large, long-lived software systems, and they are concerned with processes for organising the large-scale human effort needed to build such systems. Large software systems typically have over 100 000 lines of code. Long-lived software systems typically have useful service lifetimes of 25 years or longer. The life cycle of a software system is the period of time that stretches from initial conception to its retirement from service. Many questions arise in connection to software processes, lifecycles and methods. For instance when is it useful to build a prototype of a system, or a part of a system in order to try it out (Standish, 1998)? Automotive and aeronautical engineers frequently build and test prototypes to try out new design

concepts. In the case of software engineering, a good answer is that using prototyping reduces risk that the systems wont meet its goals or be finished on time and within budget. Besides the actual design and systems development software engineering addresses skill concerned with the management and organisation of large software projects. Concepts such as schedules, budgets, resource management policies, and risk item resolution come into play.

## *4.2 Software Architecture*

Computer systems are complex. The view of the systems structure and its content changes with the perspective of the system. As designers and developers we can talk about the description of the system or the actual execution of the system. We can reason about the system on an abstract logical level, or we can focus on the hardware and the running processes. During the design of the system architecture we have to deal with this complexity and work both on the abstract logical level and on the physical level.

The system architecture is made up of two parts, the component architecture and the process architecture. In order to produce the system architecture the designer must use several overlapping perspectives while traversing form the conceptual/logical perspective at one end to the physical perspective at the other end (Mathiasen, 1999).

While creating the component architecture the designer is focusing on the entity classes and the stable relations. The component-architecture contains a structure of the relations and connections between the different components. The decisions governing the component-architecture are mostly handled on a logical level. When designing the process-architecture the focus is on the objects and the dynamics of the system. The process-architecture is organising the processes with respect to matters such as coordination, implementation platform, efficiency and the design work with the process-architecture is usually handled on more physical level as opposed the component-architecture. By focusing on one perspective at the time the designer reduces the complexity, and by combining different perspectives and relating them to the underlying specification it is possible to create a solid comprehension of the system architecture. The borderline between different perspectives is of course not static but rather floating. While using different perspectives during the design process the system developers are still focusing on the same phenomena: the objects of the system and there internal relations (Mathiasen, 1999).

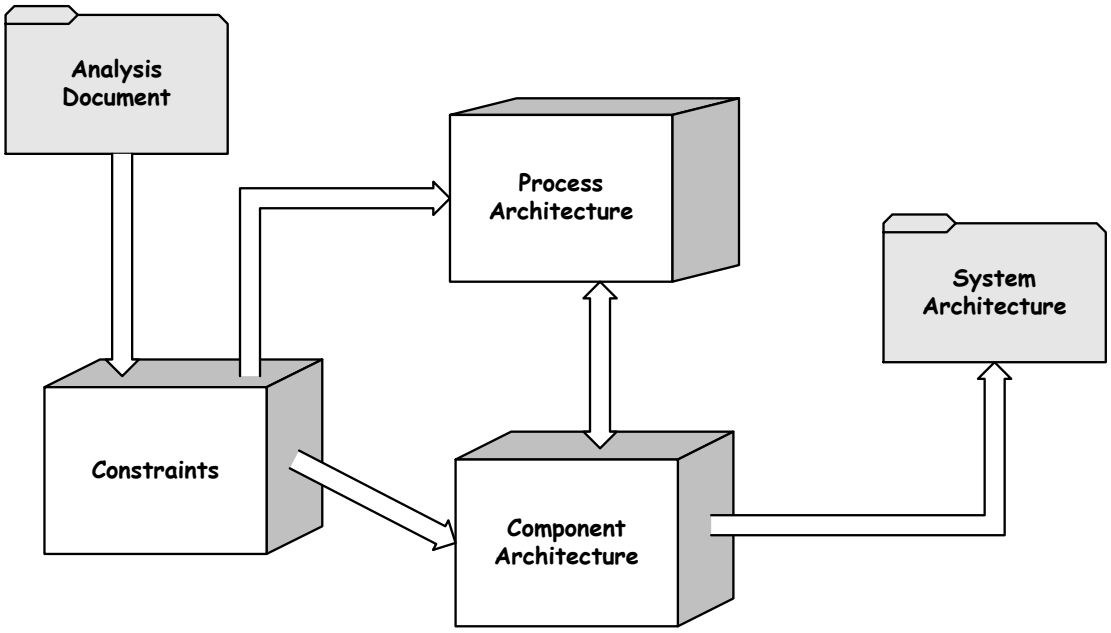



**Figure 4.1 Architectural Design**

## *4.3 Distributed System Architectures*

Virtually all computer-based systems are now distributed systems. A distributed system is a system where the information processing is distributed over several computers rather than confined to a single machine. Obviously, the engineering of distributed systems has a great deal in common with the engineering of any other software but there are specific issues that have to be taken into account when designing this kind of system.

The different components in a distributed system may be implemented in different programming languages and may execute on completely different types of processors. Models of data, information representation and protocols for communication may all be different. A distributed system therefore requires some software that can manage these diverse parts. The term middleware is used to refer to this software – it is in the middle between the different distributed components of the system (Sommerville, 2001). Middleware is general-purpose software usually bought of-the-shelf rather than written specially by application developers. Sommerville (2001) defines three different types of distributed systems: multiprocessor architectures, client-server architectures and distributed object architectures and the two last mentioned are the ones that are of interested for our purpose.

### 4.3.1 Client-Server Architectures

In client-server architecture, an application is modelled as a set of services that are provided by servers and a set of clients that use these services (Orfali & Harkey, 1998). Clients need to be aware of the servers that are available but usually do not know of existence of other clients. Clients and servers are different processes in the system as shown in figure 4.2, which is a logical model of distributed client-server architecture. There is not necessarily a 1:1 mapping between processes and processors in a system. A system containing two server computers and four client computers could run the client and server processes shown in fig 4.2.

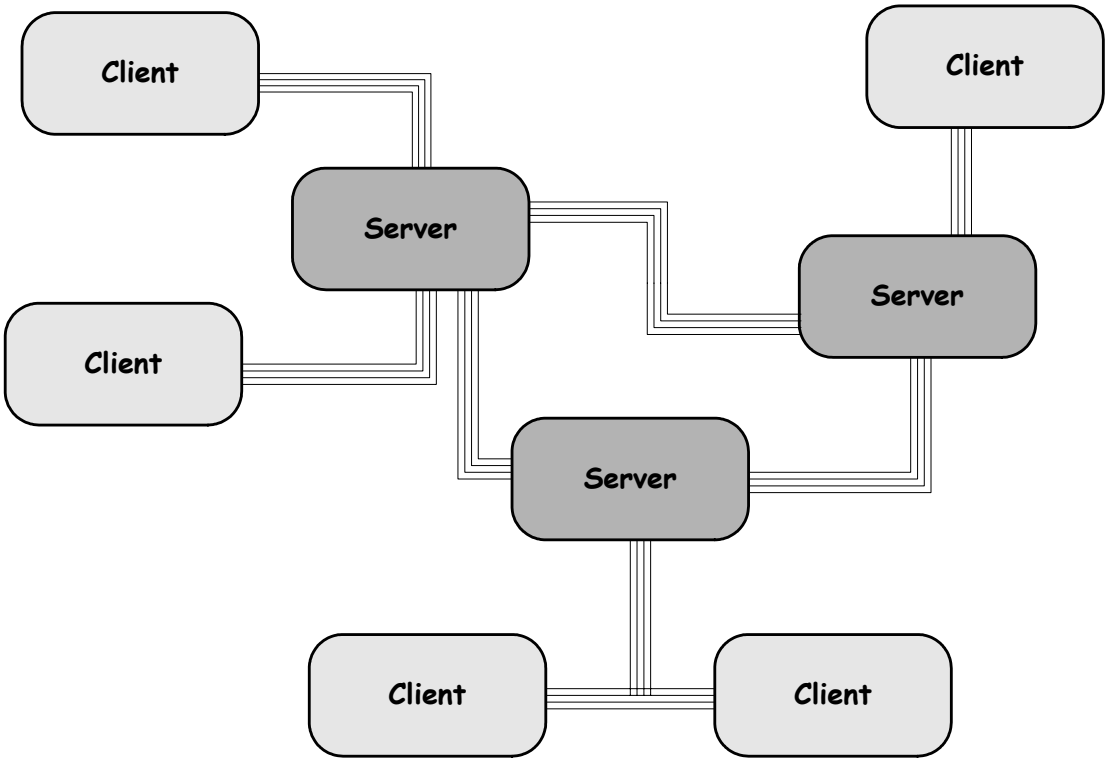



**Figure 4.2 Client Server System**

When discussing clients and servers the notion generally concerns the logical processes rather than the physical computers on which these processes execute. The design of client-server systems should reflect the logical structure of the application being devolved. One way to look at an application is to illustrated in fig 4.3, which shows an application structured into three layers (Mathiasen, 1999), (Sommerville, 2000). The presentation layer is concerned with presenting information to the user and with all user interaction. The application layer is concerned with implementing the logic of the application and the data management layer is concerned with all database operations. In centralised systems these need not be clearly separated. However when designing a distributed system, the designer should make a clear distinction between them as it then becomes possible to distribute each layer to a different computer.

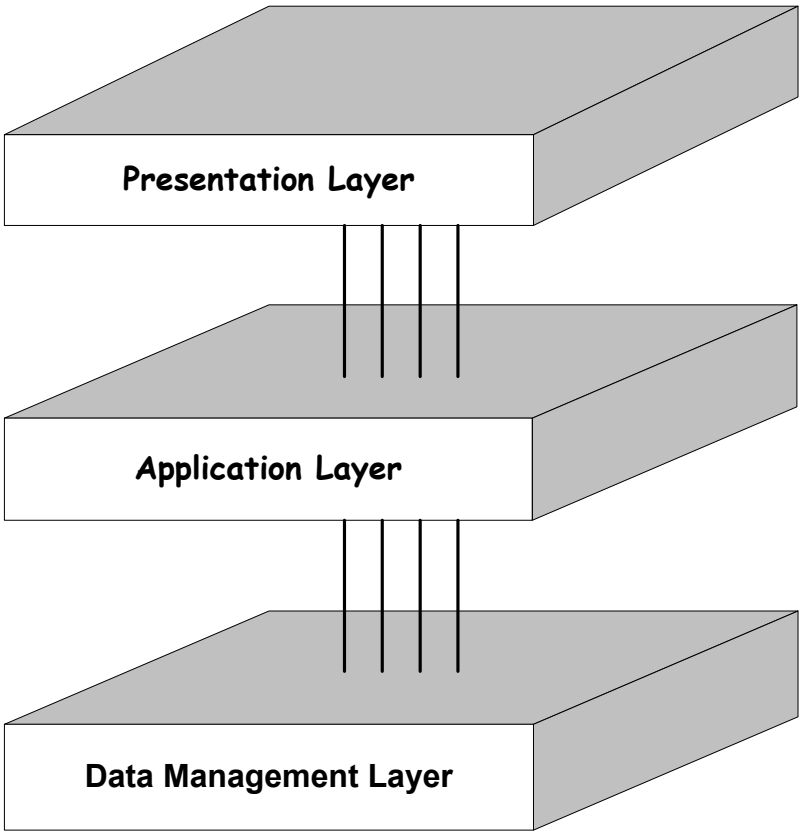



**Figure 4.3 Three-Tier Design**

The simplest client-server architecture is called a two-tier client-server architecture where an application is organised as a server or multiple identical servers, and a set of clients. The two-tier client-server architecture is either on the form thin-client model or on the form fat-client model. In a thin-client model all of the application processing and data management is carried out on the server. The client is simply responsible for running the presentation software. In a fat-client model, the server is only responsible for data management. The software on the client implements the logic and the interactions with the system user.

The final decision on component distribution should be based not only on the individual application, but also on the mix of applications operating on the system. For example an application might require extensive GUI processing and little central database processing. This would lead to the use of powerful workstations on the client side and a "bare bone server" (Pressman, 1997). With this configuration in place, other applications would favour the fat-client approach so that the capabilities of the server would not need to be upgraded.

### 4.3.2 Distributed Object Architectures

In the client-server model of distributed system, clients and servers are different. Clients receive services from servers and not from other clients. Servers may act as clients by receiving services form other servers but they do not request services from clients. Clients must know the services that are offered from specific servers and they must also know how to contact these servers. This model works fine for a lot of applications but it limits the flexibility of the system designer because they have to decide where the services are to be provided. When designing a distributed system the designer must plan for scalability, which means that it must be possible to distribute the server load as more clients are added (Sommerville, 2001).

A more general approach to distributed system design is to remove the distinction between the server and the client and design the system architecture as a distributed object architecture. In a distributed object architecture the fundamental system components are objects that provide an interface to a set of services they provide. During execution objects can call the services with no logical distinction of a receiver of a service or a provider of a service. The actual objects in this type of system can be distributed across a number of computers in a network and communicate through middleware.

One type of middleware is the Object Request Broker (ORB) (Pressman, 1997). The ORB is a middleware software component that enables an object residing on one computer to send a message to a method that is encapsulated by an object that is distributed to another machine within the network. The ORB intercepts the message and handles all of the communication and coordination activities required to find the object to which the message was addressed, call its method, pass the appropriate data to the object, and pass the resulting data back to the object that generated the message in the first place. There are several widely used standards for object request brokers e.g. CORBA developed by the Object Management Group (OMG), Suns Java RMI, Microsofts DCOM among others.

Using this type of distributed architecture allows the developer to delay decisions on where and how the services should be provided. Service providing objects may execute on any node of the network. The distinction between fat- and thin-client models becomes irrelevant, as there is no need to decide in advance where application logic objects are located. Distributed object architectures are open system architectures that allow new recourses to be added to it as required and the system becomes flexible and scalable. It is possible to reconfigure the system dynamically with objects migrating across the network as required. Sommerville (2001) identifies two strategies for using distributed object architectures.

As a logical model that allows the designer to structure and organise the system. In this case the initial focus is on providing application functionality in terms of services and combination of services. The next step concerns the distributed objects needed to provide these services. At this level the objects of the design are usually large-grain objects, sometimes referred to as business objects, which provide domain specific services.

As a flexible approach to the implementation of client-server systems. In this case the logical model of the system is a client-server model but both clients and servers are realised as distributed objects communicating through middleware. This makes it possible to change the system from for example a two-tier system two a multi-tier system. In this case the server or the client may not be implemented as a single distributed object but may consist of a number of distributed objects.

### 4.3.3 Web Services

A software service is something that accepts digital requests and returns digital responses. Using this definition, a C function, a Java object and a Structure Query Language (SQL)-stored procedure are all examples of software services. A computer application can be thought of as a set of services. Until recently a software service could only be used within a particular language or platform and was often not accessible across a network. Web services are a new breed of software component that is language, platform and location independent. They are Extensible Mark-up

Language (XML)-based building blocks for applications whose part can reside in a single machine or be distributed over vast networks. The term web services is an abbreviation for Web of Services, meaning that distributed applications will be assembled from a web of software services in the same way that web sites are assembled from a web of HTML pages (Newcomer, E. 2002).

Web services are XML-applications mapped to programs, objects, and databases or to business functions. Using a XML document created in the form of a message, a program sends a request to a Web service across the network, and, optionally receives a reply, in the form of a XML document sent as a message. Web service standards define the format of the message, and describe conventions for mapping the message into and out of the programs implementing the service. The underlying software implementations of Web services can be created using any programming language, operating system, or middleware system. Web services combine the execution characteristics of stand-alone applications with the abstraction characteristics of the Internet.

The level of abstraction at which Web services operate encompasses several interaction styles such as RPC (remote procedure call) emulation, asynchronous messaging, one-way messaging, broadcast, and publish/subscribe. Most major database management systems, such as Oracle, SQL Server and DB2, support XML parsing and transformation services, allowing direct interaction between Web services and the database. Middleware vendors typically also provide a mapping of Web services to their software systems, such as application servers and integration brokers. Web service technologies usually encompass two interaction patterns; remote procedure call and document oriented.

In RPC-oriented interaction, the Web service request takes the form of a method or a function call with associated input parameters and returned data. In contrast to the document-oriented interaction, the RPC-oriented interaction sends a document formatted specifically to be mapped to a single logical program or database. In the document-oriented interaction style, the Web service request takes the form of a complete XML-document that is intended to be processed whole. This is like submitting a message to a queue for asynchronous processing.

## *4.4 Information Exchange*

Companies and organisations have been using computers for information processing since the early fifties but it was not until the early 1970s that the use of computer based information systems became common. Data exchange between remote computer systems that belong to different organisations is called electronic Data Interchange EDI. One of the main obstacles when it comes to globalise the construction industry is the lack of a standard language for data exchange. Product and process model data standards are required in order to produce an information exchange language for the construction industry (Pouria, 2001).

### 4.4.1 Product Data Standards

Several research projects have developed building data models that could be used as standards within the construction industry. Two of the major initiatives when it comes to standardising product models are the International Organisation for Standards ISO STEP standard 10303 for product data representation and exchange and the International Alliance for Interoperability IAI (Zarli, 2002). The objective of the STEP standards is to represent information about products in different industries. The aim is to facilitate a notion to be able to describe product data independent from any specific system throughout the life cycle of the product.

STEP technology is practiced in several industries including the automotive and the electronics industries. The building Construction Core Model BCCM part 106 is intended to be used as a unifying reference for building construction applications protocols. These standard models should be used using the STEP physical file format for simple file exchange or STEP Standard Data Access Interface SDAI for online file exchange.

The IAI is a global consortium for architecture, engineering, construction and facility management that develops standards for construction projects called the Industry Foundation Classes IFCs. The intention of the IFCs is to define how real things such as doors, walls, etc. and abstract things such as spaces and processes should be represented electronically. IFCs build on result from STEP and an increasing number of software products support IFCs (Pouria, 2001). Efforts in recent years through ISO and the IAI have been focusing on defining the syntax and semantics of a standard language. The IFCs and the more recent aecXML (aecXML, 2002) could be the basis of the future development of the common language for data exchange in the construction industry.

### 4.4.2 Design for Dynamic Virtual Environments

Zarli (2002) suggests the following design for a virtual environment within the construction domain. In order to deal with any type of client, connected to a network a loose coupling should be provided that allows the enterprise logic to be independent from any presentation and client issues. A fundamental issue is the information access and transfer between the different layers of the proposed design and between all the components that form each of these layers. Zarli (2002) identifies four types of data transfer.

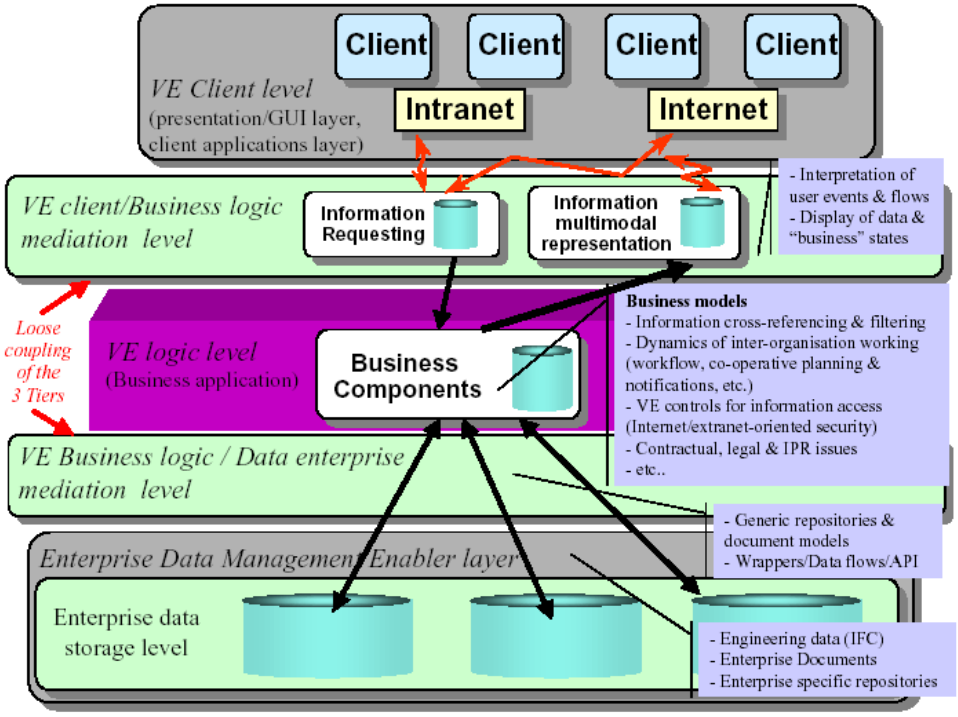



**Figure 4.4 Multi Tier Design (Newcomer, 2002)**

*Model-based transfer:* when the client wants to receive information the whole model e.g. a full STEP model is transferred from the server to the client. This approach could be implemented using the STEP SPF technology.

*Object-based transfer:* when the client wants to access data that is not within the client yet, only the referred objects are transferred. This approach minimises the information transfer and this transfer type could be implemented using XML.

*Method-based access:* whenever the client needs data, it calls the appropriate remote method on a remote object. The called method is declared within an interface for the object in question.

*Operations-based transfer:* this specific design consists in not transferring the requested data but actually transferring the operations required to rebuild the whole set of data on the client side. This approach is useful when the objective is to avoid the transfer of very large sets of data.

In order to decide upon a certain technology to implement the designer must assess the different end-user requirements and the technical constraints of the proposed technologies. Each of theses technologies have there own characteristics. Criteria for selection could for instance be: management of messages, protocol issues, shipping of large set of data versus small set of data, data duplication and control of the coherency and synchronisation issues.

### 4.4.3 XML a General Overview

SGML (SGML, 1986), the Standard Generalised Markup Language is the dominating reference when it comes to dealing with structured documents. SGML is a language that describes how tags can be inserted into a document. With the evolution of the Web, SGML-like document repositories have been distributed. XML have been developed by the W3C, the World Wide Web Consortium, in order to present a less complex standard compared to SGML, for the exchange of structured documents within Intranets or the Internet. XML is text-based, self-describing and becoming accepted as a global standard. It is a meta-language that can be used to create any new XML-based language and it is essentially a data presentation language that can be used for displaying, exchanging and storing data. In a certain domain it is possible create new presentation semantics, that is unaffected by the actual content of the XML message.

XML is a universal, easy to comprehend and very adaptable file format, well adapted to online catalogues and product configuration information. The XML technology have various features that supports large scale Web content providers for industry, media-independent publishing, vendor-neutral data exchange, workflow management in collaborative environments, and the processing of Web documents by intelligent agents. Compared to HTML, which only enables to markup the information for presentation, XML provides a standard way of structuring data within different information systems and essentially Web based information systems. XML is at the moment gaining acceptance within two main areas: cataloguing and further distributing and recovering information on the client side, and data exchange and application integration at the enterprise level services. XML can be viewed as a standard way of passing data between many heterogeneous distributed application servers, as well as across multiple operating systems.

### 4.4.4 XML for Information Exchange and Applications Integration

XML is both a technology for the manipulation of structured documents, and a technology for a flexible and dynamic representation of complex objects. Since XML documents can be exchanged using any underlying protocol, XML can be used to define the container for a message content for any type of data. Regarding data exchange and interoperability XML can be used between data warehouses and the middle layer in a 3-tier architecture, for data integration and linking through documents conveyed or generated from multiple sources. Further it can be used between the middle layer and the clients for data delivery, or delivery to other applications for further processing, data manipulation and for the creation of multiple views e.g. to facilitate the use of multiple clients. XML is a platform-independent meta-language that offers means for data serialisation. Using XML in conjunction with java offers both code portability and platform independence (McLaughlin, 2001), (Zarli, 2002).

### 4.4.5 XML – STEP Technology

As a message-centred technology, XML is adapted to the exchange of information allowing delivering of various formats of documents presentation or message structuring. The STEP standard especially through EXPRESS, offers a technology for the description and representation of product information including their semantics and a lot of normalised models (Celander, 2001). When comparing STEP with XML regarding information exchange Zarli (2002) presents the following points in favour of XML.

Step-.files requires the exchange of a complete STEP model, whilst with XML it is possible to exchange parts of information when this is required.

Transmission of information can done over the Internet e.g. by using the HTTP-protocol which allows the crossing of fire walls, and this is not the case with STEP

A lot of XML parsing tools already exist, and some of them are free software, while the parsing of STEP-files requires specific developed parsers.

XML messages do not vehicle semantics for their contents and this is also the case for STEP-files, which need to be transferred with their corresponding EXPRESS schema. A fundamental issue with XML is that it does not allow specifying a given semantics for each tag used to structure documents. There is no automatic way in XML to specify common semantics for the tags <*facility*> and <*equipment*> manipulated in distinct documents generated from separate applications. With XSL, it is simpler to create conversion between tags so as to convert a document using one DTD in another document using another DTD, provided that it is possible to relate tags between themselves. In order to avoid such conversions, which could be costly at runtime, the best thing is to agree on a common DTD (Zarli, 2002). The objective is to guarantee a common understanding of the message structure before any operation. Within in the industry several organisations are investigating or already defining such agreements and essentially two approaches are emerging: exchange between a lot of applications through common agreed message formats by the definition of a single DTD and exchange and XSL-based transformation whenever required between few applications and preferably when the frequency is low.

According to Zarli (2002) STEP and XML must be viewed as complementary technologies, and he further argues that it appears valuable to specify and develop gateways allowing exploiting STEP-based information through dedicated technologies like XML dealing with heterogeneous and adaptive environments. Within the construction industry different work recently emerged like aecXML (aecXML, 2002) and bcXML (bcXML, 2002). These works tend to define DTDs and XML-based mechanisms for STEP-based information exchange. A future possibility is for STEP to concentrate on semantics and structuring of product information like data for the whole product lifecycle. STEP can then benefit from the technological development around message brokering and the Internet especially XML, which

provides a logical separation between semantics on one side and data access, transfer and presentation on the other side.

## *4.5 Mobile Computing Design*

Mobile computing differs from desktop computing in several aspects. Mobile devices, e.g. PDAs, mobile phones and digital cameras have, compared to desktop computers low computational power, small memory and often no mass storage. Communication links to other mobile devices or to a stationary network are usually wireless and often with lower bandwidth compared to desktop computers hooked up to LAN. This section presents a conceptual framework for mobile IT use proposed by Kristoffersen and Ljungberg (1998) and patterns of mobile interaction proposed by Roth (2001).

### 4.5.1 Mobile IT Use

According to Kristoffersen and Ljungberg (1998) we are all mobile since we are to be considered mobile whenever we are in motion. People are in motion on a daily basis, commuting to work, visiting friends etc. But we are also still on a regular basis and these periods of time may be considered as moments of non-mobility. Instead of trying to define mobility one need to look at the different situations when people are or are not mobile. One approach is to look upon mobility within different settings. Kristoffersen and Ljungberg use three modalities wandering, travelling and visiting.

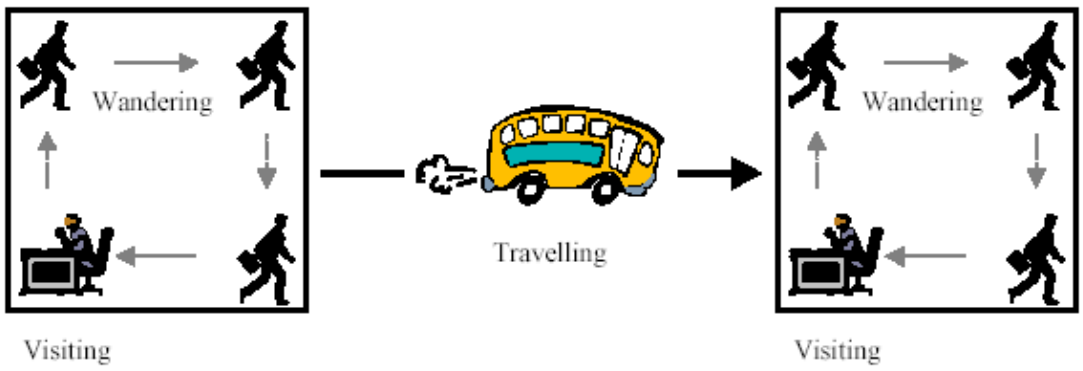



**Figure 4.5 Modes of Modality**

Wandering is an activity where the individual is mobile within a defined area like a factory or an office building. A wanderer may be a service technician at production line, wandering around solving problems.

Travelling is an activity where an individual is travelling from one place to another. The traveller may be symbolised by a commuter taking the subway to work.

Visiting is an activity where an individual is visiting a specific building or geographic location for a limited period of time. A consultant that works at client's office could be considered a visitor.

Below is Kristoffersen and Ljungberg's model of mobile IT use. Its three main components are environment, modality and application. Environment is the users physical and social surroundings. Modality is the fundamental patterns of motion as described above. While application is the combination of technology, program and data used.

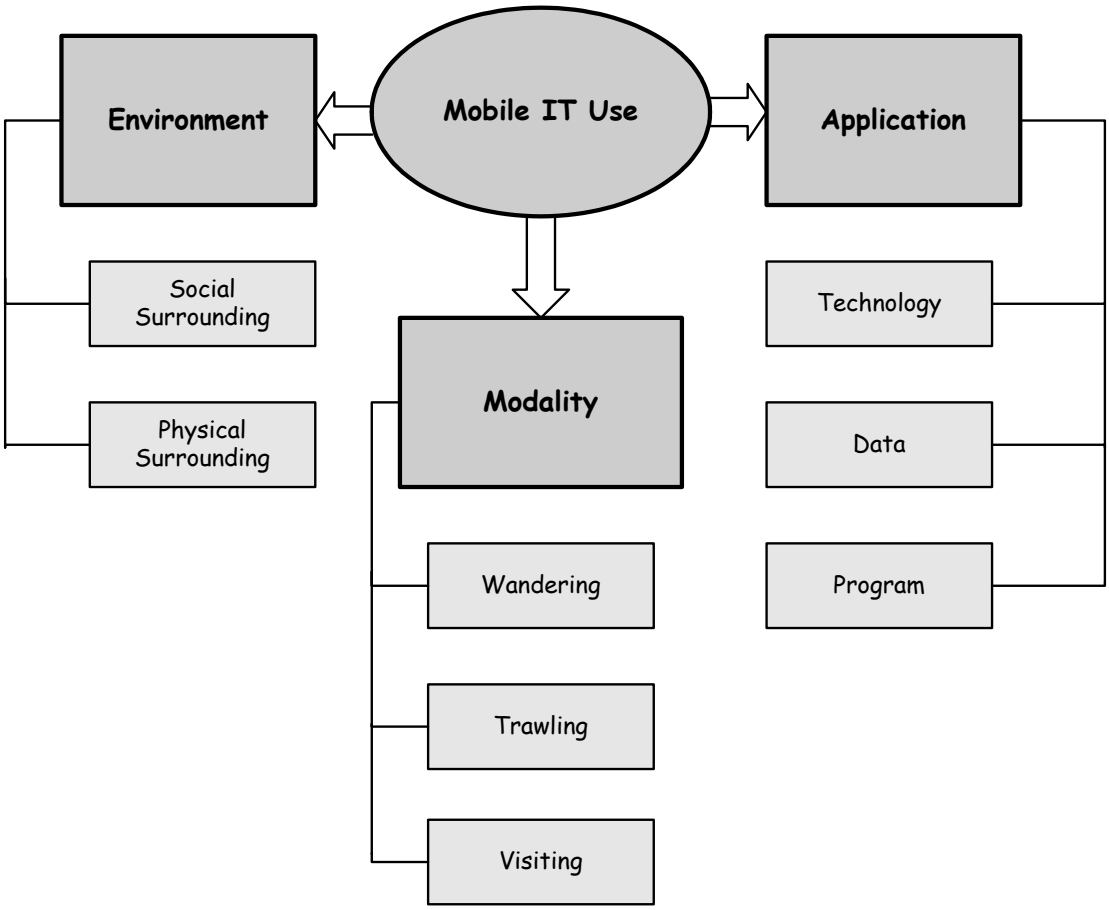



**Figure 4.6 Mobile IT Use**

### 4.5.2 Patterns of Mobile Interaction

When there are multiple users and multiple devices in a system design the architect has to ponder several problem areas like, distribution of application on available machines, network availability and bandwidth, security issues (e.g. privacy, integrity, authenticity) and the constraints regarding the actual user interface (e.g. screen size, interaction stile). One method to address design tasks is the use of *design patterns*. Patterns are derived from successful software designs and can be reused as building blocks for new designs. Roth (2001) presents the concept of mobility patterns in order to cover problem areas within the field of mobile computing. According to Roth (2001) related patterns can be grouped together to pattern classes using a pattern hierarchy

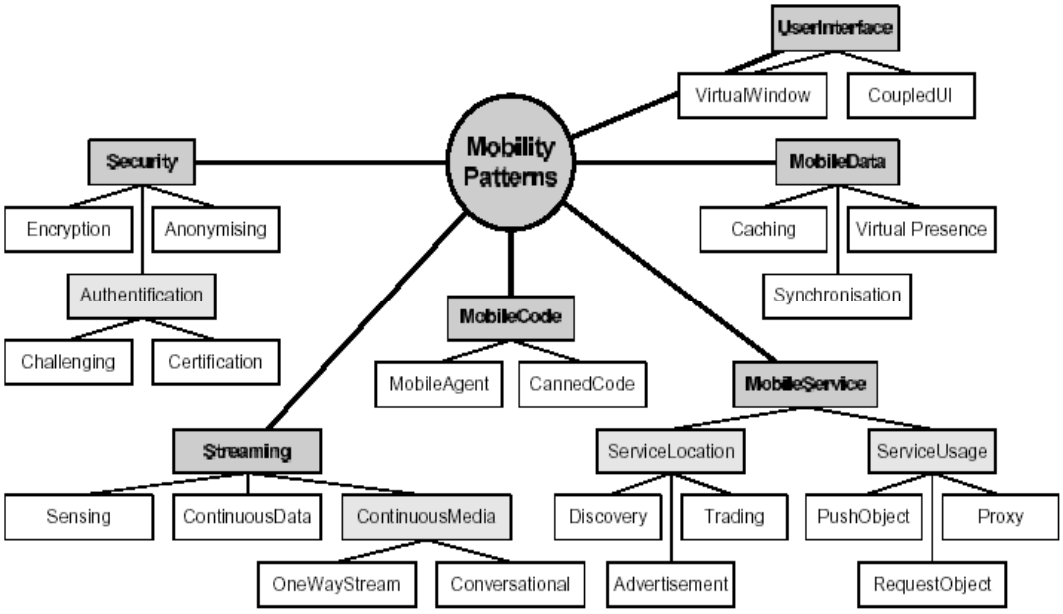



**Figure 4.7 Mobility Patterns**

A pattern can be described via a set of characteristics like synopsis, context, forces solution, consequences, examples, related patterns and classes. The class characteristic indicates how the pattern is integrated in the class hierarchy. Roth (2002) gives two examples of mobility patterns, *The Synchronisation Pattern* and the *Proxy Pattern*.

The synchronisation pattern address the problem with identical data stored on different devices, which are weakly connected. Several users apply data changes to different devices and these data updates often occur simultaneously. For example consider two users carrying two identical databases stored in their PDAs, which have been originally copied from a central server. While travelling, they are only

really connected to their home database. Identical data stored at different locations tend to run out of sync when users apply modifications on their locally stored data specially when the device itself is rarely connected.

The proposed solution to the case above is to keep track of modifications applied to local data, exchange modifications with corresponding databases on other devices whenever they reconnect and to detect data conflicts and further implement a conflict resolution strategy. This pattern can be used when data stored on different devices has not to be strongly up to date. However, users should be aware that other users could change the same data simultaneously, which may cause conflicts later. Since connected devices have to compare their local changes, only data that can be stored I tables and lists, can be used with this pattern. Weakly structured data can be compared record by record, which increases the probability for unwanted conflicts. This pattern is not suited for continuous data such as audio or video. SyncML (SyncML, 2002) is a framework for data synchronisation in mobile scenarios.

The proxy pattern, addresses the problem when a device has not capability to perform a requested task, and connects to another device with higher computational power in order to perform the requested task. Consider a user browsing the web with a handheld device. The screen resolution of a handheld device is currently poor and advanced graphics can be difficult to display. Rendering complex elements requires a lot of computational power, which are often not available on a handheld device. For example a user who requests a specific task and expects a certain amount of network bandwidth, a certain amount of computational power on the local device and also expects a certain level of interaction behaviour. The device being used is not capable of performing the requested task according to the above mentioned conditions.

The solution to this scenario means that the device does not connect directly to the required service end point; instead it calls another device or computer to perform these tasks. This other device or computer is called the proxy. The proxy accepts service request from other devices, connects to the actual service provider and performs the task, processes the results and finally, sends them back to calling device.

This pattern is a general pattern and useful in various mobile scenarios. Very often, the devices used by end-users have constrained capabilities when compared to stationary devices. When end-users want to execute demanding task there are two crucial points in this pattern. The proxy itself, in case of failure, task execution is disabled, even if the requesting device and the service provider are on-line. The second crucial point concerns the communication link between the requesting device and the proxy. If this link is broken, the task cannot be performed, even if the proxy has successfully executed the task. It is clear that the proxy in the general case must have more or other capabilities compared with the end-user device. As the proxy provides a specific service, a mobile device must be able to find the proxy inside the

network. There are two approaches in order to find the proxy. A proxy has to have a fixed network address or the mobile device can with the help from a service discovery mechanism resolve the network address.

ProxyWeb (Intellisync, 2002) allows a handheld user to browse the web witout struggling with device limitations. The proxy pre-processes web pages, downscales graphics and pre-computes the appropriate layout. As a result the amount of data transferred to the handheld device is drastically reduced and the devices are relived from heavy rendering tasks.

# 5 General Mobile Application Server Design

This chapter describes our design and to some extent the design of the different third party frameworks used within it. This design chapter differs from traditional design presentations since the design presented to such a large extent is constructed by combining third party frameworks. Also, this chapter do not present any specific products, for information on software that may be used to implement the design below see the next chapter, Specified Mobile Application Server Design.

## *5.1 Design Overview*

The *Mobile Application Server* is designed to be the gateway between enterprise systems and mobile clients, offering mobile users an integrated, personalised and client adapted view of the connected systems. To make this possible the server needs to be easy to integrate in the present IT infrastructure and be mobile client independent. As one mean of making this possible, industry standards are used for all communication between the server and its surroundings. Open standards are also used within the server, specifying the interfaces between the different parts of the design. This makes the server highly modular, making it possible to easily replace parts of the server over time in response to shifting demands.

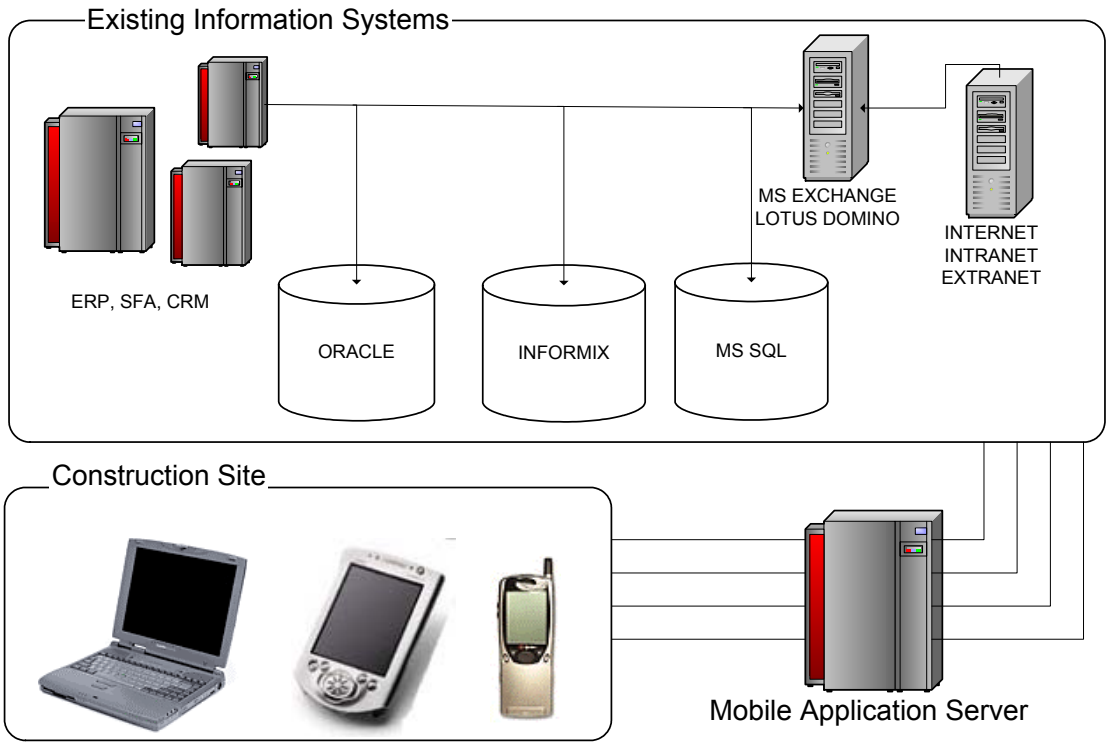



**Fig 5.1 Mobile Application Server Design Overview**

To understand the concept of a *Mobile Application Server* it is necessary to be aware of the problem that the server tries to solve. On the left in the figure below there are some examples of services that are commonly used within companies today. These services are most often not designed for mobile use and as the demand for mobile computing rises there is a need for gateways that bridge the gap between the present services and the mobile users. The new mobile clients differ from traditional desktop clients with regard to, among other things, performance, screen size and connectivity

Below is a figure illustrating the different parts of the *Mobile Application Server* and how they are combined. The possibility to integrate the server with already present IT infrastructure is, as already stated, of outmost importance. The design supports three different types of integration solutions. One is the J2EE Connector architecture (JCA), which enables an enterprise information systems vendor to provide a standard resource adapter for its system. The resource adapter plugs into the application server, providing connectivity between the enterprise system and the application server. Another integration approach is the use of enterprise messaging, which is suitable for loosely, coupled, asynchronous interaction between systems. Enterprise messaging is supported by the use of the Java Messaging Service (JMS), which is presented later in the enterprise java technology overview together with other enterprise java technologies as JCA. It is also possible to use XML-messaging for integration, either by the use of web services or by using a custom way of exchanging XML-messages. The use of web services could also be considered a fourth way of integration but in this design it is regarded as merely another form of XML-messaging for simplicity.

The *Mobile Application Server* business logic is encapsulated in Enterprise Java Beans (EJB) and servlets. The data is stored in a relational database and a XML native database, which database to use in each case depends on the nature of the data. Java Server Pages (JSP) and an XML framework, the same framework that is used for integration, is used to describe the presentation logic that defines how data should be presented. Finally, there is also an additional layer with adaptation logic that handles personalisation, client adaptations and other features that depends on who is using the system and by which means.

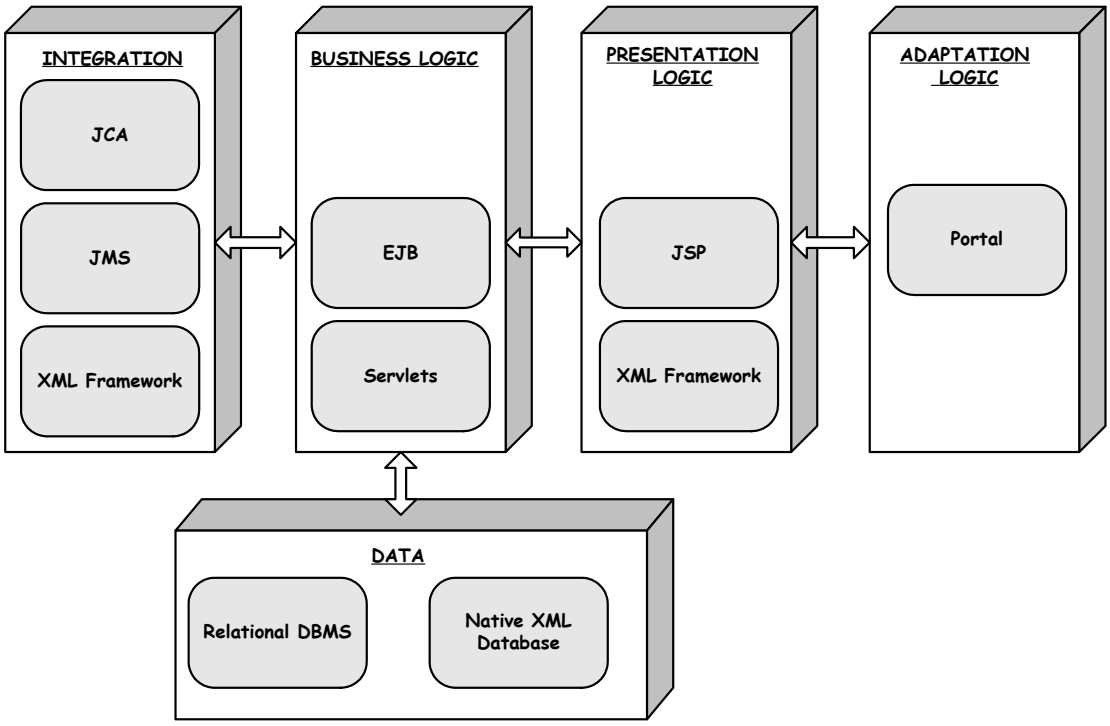



**Figure 5.2 Mobile Application Server Design**

## *5.2 Server Architecture*

Our design is based upon the Java 2 Enterprise Edition (J2EE) architecture. The decision to use J2EE was made in order to make it possible to handle multithreading, synchronisation, transactions, and resource allocation management in an efficient manner. Other issues that made us turn to the J2EE platform were scalability, maintainability and the support for legacy system integration. These advantages come at a price though; the J2EE framework adds some additional complexity to the design process.

J2EE is a pure Java framework and therefore it may be used on all common platforms and operative systems. To be able to apply the framework an application server that implements the J2EE specifications is needed. An application server is an application that provides the basic infrastructure required for developing and deploying multitiered enterprise applications (Alur et al, 2001). In the past application servers from different vendors were not compatible and an application developed for one server could not be easily moved to another. Today most leading application server vendors have implemented the J2EE specifications, which in theory makes true portability a reality. But since its common practice among vendors to supply complementary vendor specific functionality the portability matter still exists. Our design uses only standard J2EE functionality to preserve vendor independence, with regard to the application server.

The J2EE framework is not vendor independent in the traditional sense since Sun Microsystems owns the rights to the J2EE specifications. But considering that the specifications have been implemented by a large number of application server vendors and open-source projects we argue that the platform is indeed vendor independent in a practical sense.

This section describes the J2EE framework and gives some pointers on how it should be used efficiently.

### 5.2.1 Architecture Overview

Multitiered architectures have become standard when designing distributed systems. The clear distinction between the layers makes it possible to distribute different layers to different computers. An overview of multitiered architectures is presented in the theoretical framework chapter (4.3). The multitiered architecture is central to J2EE and it consists of the following tiers:

Client tier – The client tier interacts with the user and displays information from the system to the user. From a developer's point of view, a J2EE application can support many types of clients. J2EE clients can run on laptops, desktops, palmtops, and cell phones. They can connect from within an enterprise's intranet or across the World Wide Web, through a wired network or a wireless network or a combination of both.

They can range from something thin, browser-based and largely server-dependent to something rich, programmable, and largely self-sufficient.

Web tier – The Web tier generates presentation logic and accepts user responses from the presentation clients, which are typically HTML clients, Java applets, and other Web clients. This tier handles all of a J2EE application's communication with Web clients, invoking business logic and transmitting data in response to incoming requests. In the J2EE platform, servlets and JSPs in a Web container implements the Web tier.

Business tier – This tier handles the core business logic of the application. The business tier provides the necessary interfaces to the underlying business service components. The business components are typically implemented as EJB components with support from an EJB container that facilitates the component life cycle and manages persistence, transactions, and resource allocation.

EIS tier – This tier is responsible for the enterprise information systems, including database systems, transaction processing systems, legacy systems, and enterprise resource planning systems. The EIS tier is the point where J2EE applications integrate with non-J2EE or legacy systems.

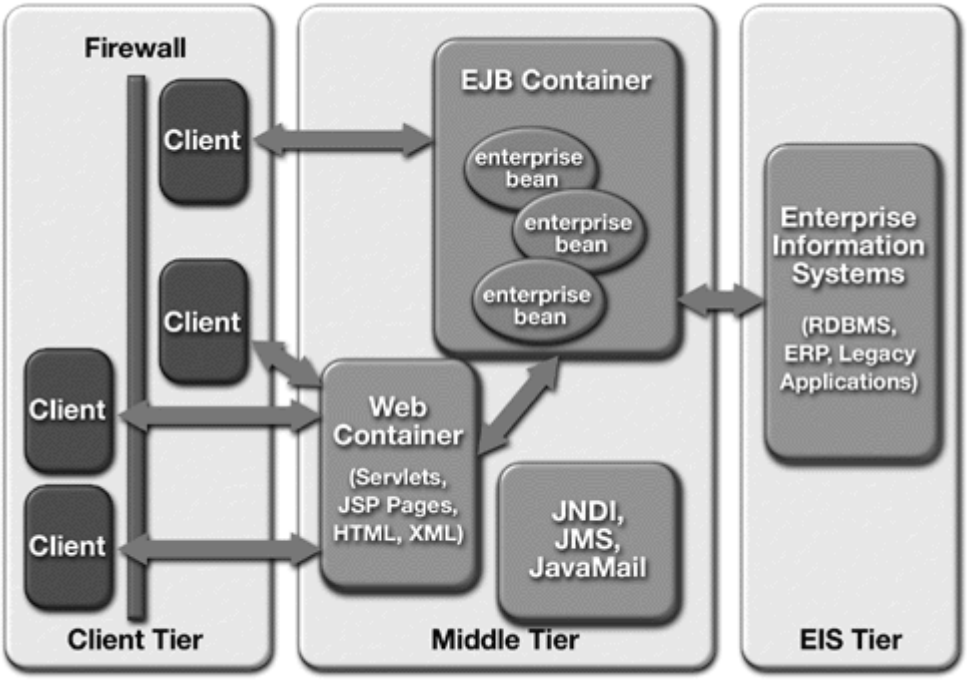



**Figure 5.3 J2EE Architectural Overview, Sun Microsystems**

### 5.2.2 Enterprise Java Technologies

The Java 2 Enterprise Edition is not one single technology but a collection of several Java technologies complementing each other. All are not used within the design but most are used directly or indirectly at some time. Below is an overview of the different technologies used in J2EE; a basic understanding of these is needed to be able to understand the thoughts behind the design presented.

Java Servlets may be seen as the server equivalents of applets. They are the Java platform technology of choice for extending and enhancing web servers. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Moreover servlets can access a library of HTTP-specific calls.

Java Server Pages (JSP) uses XML-like tags and scriptlets written in the Java programming language to encapsulate the logic that generates the content for the page. This separates the user interface from content generation enabling designers to change the overall page layout without altering the underlying dynamic content. Java Server Pages technology is an extension of the Java Servlet technology.

Enterprise Java Beans (EJB) is one of the most important technologies within the J2EE framework and it is used extensively within the design presented. In the section following this overview EJB is presented in greater detail.

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting the J2EE platform to heterogeneous EISs. Examples of EISs include ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language. By defining a set of scalable, secure, and transactional mechanisms, the J2EE Connector architecture enables the integration of EISs with application servers and enterprise applications.

The J2EE Connector architecture enables an EIS vendor to provide a standard resource adapter for its EIS. The resource adapter plugs into an application server, providing connectivity between the EIS, the application server, and the enterprise application. If an application server vendor has extended its system to support the J2EE Connector architecture, it is assured of seamless connectivity to multiple EISs. An EIS vendor needs to provide just one standard resource adapter, which has the capability to plug in to any application server that supports the J2EE Connector architecture.

The Java Naming and Directory Interface (JNDI) is a standard extension to the Java platform, providing Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. As part of the Java Enterprise API set, JNDI enables seamless connectivity to heterogeneous enterprise naming and directory services.

Java IDL adds CORBA (Common Object Request Broker Architecture) capability to the Java platform, providing standards-based interoperability and connectivity. Java IDL enables distributed Web-enabled Java applications to transparently invoke operations on remote network services using the industry standard OMG IDL (Object Management Group Interface Definition Language) and IIOP (Internet Inter-ORB Protocol) defined by the Object Management Group. Runtime components include an Object Request Broker (ORB) for distributed computing using IIOP communication.

JDBC technology is an API that lets a developer access almost any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL databases, and now, with the new JDBC API, it also provides access to other tabular data sources, such as spreadsheets or flat files.

The Java Messaging Service (JMS) API enhances the J2EE platform by simplifying enterprise development, allowing loosely coupled, reliable, asynchronous interactions among J2EE components and legacy systems capable of messaging. By combining Java technology with enterprise messaging, the JMS API provides a powerful tool for solving enterprise-computing problems. Enterprise messaging provides a reliable, flexible service for the asynchronous exchange of critical business data and events throughout an enterprise. The JMS API adds to this a common API and provider framework that enables the development of portable, message based applications in the Java programming language.

The Java Transaction (JTA) API specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications.

The Java Transaction Service (JTS) specifies the implementation of a Transaction Manager. A JTS Transaction Manager provides transaction services to the parties involved in distributed transactions: the application server, the resource manager, the standalone transactional application, and the Communication Resource Manager (CRM).

Java Remote Method Invocation (RMI) technology run over Internet Inter-Orb Protocol (IIOP) delivers Common Object Request Broker Architecture (CORBA) distributed computing capabilities to the Java platform. Java RMI over IIOP combines the features of Java RMI technology with the features of CORBA technology. Like Java RMI, RMI over IIOP speeds distributed application development by allowing developers to work completely in the Java programming language

The JavaMail API supports reading, composing, and sending electronic messages. It’s used to create Mail User Agent (MUA) type programs, similar to Eudora, pine, and Microsoft Outlook. The API's main purpose is not for transporting, delivering,

and forwarding messages; this is the purview of applications such as sendmail and other Mail Transfer Agent (MTA) type programs. MUA-type programs let users read and write e-mail, whereas MUAs rely on MTAs to handle the actual delivery.

JavaBeans Activation Framework (JAF) offers developers standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and to instantiate the appropriate bean to perform said operations. JAF is used by JavaMail and other tools that need to handle different MIME-types.

### 5.2.3 Enterprise Java Beans

The EJB container corresponds to the business tier and contains the business components. A business component within the J2EE framework is called an enterprise bean. These beans are usually of two basic types, entity beans that represent business entities and session beans that represent business tasks, however a third type of enterprise bean as emerged the message driven bean. Another way to describe the difference between the standard bean types is that entity beans are used to represent data while session beans normally are used to access data. As the entity beans embody the business data they must be persistence (i.e. the entity beans must be stored in database or equivalent). The persistence is handled by the EJB container, which creates, disposes and manages the beans. The EJB container also offers secure means of communication between beans and other parts of the J2EE architecture.

When defining an entity bean one may choose to define how the bean should be stored in the database or one may let the container handle this by its own. The first approach is called Bean Managed Persistence (BMP) and used when the developer wishes to have complete control over how an entity is stored. The second approach is called Container Managed Persistence (CMP) and should be the standard way to handle entity bean persistence as it has several advantages. The time to market may be shortened since less code needs to be written and maintained. The portability is improved since there is no need to write vendor specific code to increase performance, the container optimises the queries for the database that are in use. In most cases performance is improved as well because of the automatic optimisation. Another positive aspect is that CMP entity beans inherit the rich relationship semantics, referential integrity, cardinality, relationship management, and cascading delete that the container provides automatically. With BMP entity beans, on the other hand, the bean developer must provide referential integrity checks and relationship management when implementing inter-entity relationships; and that's no trivial task (Tulachan, 2002).

But of course there are some drawbacks with CMP, the developer losses control and the learning curve is steep for many developers since they need to use new query languages instead of the familiar SQL. An effect of the loss of control may be increased debugging difficulties.

Before CMP was introduced one had to handle relations among entity beans manually, but now this may be left to the container. This is called Container Managed Relations (CMR) and as stated above it may simplify the handling of inter-entity relationships. However dependencies between entity beans should be avoided as much as possible, since such dependences create overheads that may impede overall application performance (Alur et al, 2001).

In general as much as possible of the business logic should be in the entity beans, using them just as mean of storing data is not good practice. But when the business logic introduces inter-entity-relations it should be moved to a session bean. Under some circumstances the entity beans should be merged instead, this is when one entity is dependent on another. The life cycle of a dependent object is tightly coupled to the life cycle of a coarse-grained object. A client may only indirectly access a dependent object through the coarse-grained object. Dependent objects may not exist by themselves; they are dependent of a coarse-grained object (Alur et al, 2001).

A session bean is created for a client and in most cases exists only for the duration of a single session. A session bean performs operations on behalf of the client such as database access or performing calculations. They may be of two types, stateful or stateless, either possessing internal states or not. If one has the option to choose, the stateless session bean should be used for better performance. The stateless beans may be pooled and used by multiple clients while stateful beans for obvious reasons is client specific.

When designing object-oriented systems it is very common to create use case scenarios. A use case describes how a user is supposed to perform a specific task, like making a withdrawal from a bank account. The session beans are well suited to represent these use cases. But each bean creates overhead and related use cases should be combined into a single session bean (Alur et al, 2001). To continue on the bank analogy, withdrawal, disposal and checking balance etc. is best represented by a single session bean.

A relatively new type of enterprise bean is the Message Driven Bean (MDB). As the name reveals, this type of bean is responsible for receiving and processing messages. The messages are asynchronous JMS (Java Messaging Service) messages; JMS is introduced in the technology overview presented earlier in this chapter. One of the most important aspects of message-driven beans is that they can consume and process messages concurrently. This capability provides a significant advantage over traditional JMS clients, which must be custom-built to manage resources, transactions, and security in a multithreaded environment. The message-driven bean container provided by the server manages concurrency automatically, so the bean developer can focus on the business logic of processing the messages. The MDB can receive hundreds of JMS messages from various applications and process them all at the same time, because numerous instances of the MDB can execute concurrently in

the container (Monson-Haefel, 2001). MDBs may perhaps be used to improve the performance of our design.

All enterprise beans except the MDB have three interfaces, these are Home, Remote and Local. The MDB lacks component interfaces because it is not accessible via the Java RMI API; it responds only to asynchronous messages. The Home interface describes the methods that handle the beans life cycle. The Remote interface contains the methods used by a client that interacts with the bean. While the Local interface provides this functionality when interacting with other beans within the container.

When designing EJBs it is important to consider performance. One should not optimise every bit of code, but when it is obvious that a piece of code will run often optimisation is generally a good idea. Most important though is not to optimise but to learn to write code in a style that performs well. Albert Einstein's famous quote "Things should be made as simple as possible, but not any simpler." may be used as a general guideline. Following this guideline simplifies development, enhances maintainability and often result in better performing software. Other similar valuable guidelines are to use the simplest java class possible and not to reinvent the wheel, developing anything yourself that you can get made for you (Halter & Munroe, 2002).

## *5.3 XML Publishing Framework*

XML has become the de facto standard for exchange of information over the Internet and therefore good XML support is vital to any major Internet based system. A construction industry server platform for mobile computing needs a robust XML framework to handle the heterogeneously stored data in an efficient manner. Currently there are a number of frameworks available, which support tasks as generation, transformation, aggregation and parsing of XML data. But this is relatively new technology and many frameworks are not yet stable enough for enterprise applications. This section describes the characteristics of a XML framework and how XML as well as non-XML data may be used and produced by this type of framework. It does not describe XML fundamentals or different XML dialects, which is presented within the theoretical framework.

### 5.3.1 XML Publishing Framework Basics

There is no common widespread definition of what makes an XML publishing framework. The description below is the authors' view of what should be essential for an XML publishing framework.

The two most essential features of an XML publishing framework is its ability to perform transformations and aggregations. Where transformations means the ability to transform an XML document, for example going from one XML dialect to another. And, aggregation simple mean the ability to combine information from two or more XML documents into one in a structured way. These requirements indirectly create a

need for the possibility to make queries in the XML documents, which can be accomplished by implementing query languages as XPath and XQuery.

Further, the framework must be able to connect to the generators of XML data in some straightforward manner. One could imagine a framework without this possibility, just taking data from a database or other form of data depository. But this would significantly reduce the frameworks usability since its common that data is produced for a specific request.

Additionally, a XML publishing framework should be able to output data in common non-XML presentation formats as html or be able to connect to software that performs the transformation from XML to non-XML formats.

Since handling transformations is one of the frameworks most crucial features it is described in somewhat more detail than the other features. The standard language for transforming XML documents into other XML documents is XSLT. A transformation expressed in XSLT describes rules for transforming a source tree into a result tree. The transformation is achieved by associating patterns with templates. A pattern is matched against elements in the source tree and a template is instantiated to create part of the result tree. The result tree is separate from the source tree and its structure can be completely different. In constructing the result tree, elements from the source tree can be filtered and reordered, and arbitrary structure can be added. A transformation expressed in XSLT is called a stylesheet. This is because, in the case when XSLT is transforming into the XSL (Extensible Stylesheet Language) formatting vocabulary, the transformation functions as a stylesheet (Clark, 1999).

### 5.3.2 Handling non-XML data

When data is not provided, or should be outputted, in a non XML-format there is usually a need to complement the framework with supplementary software. This section presents how GIS and CAD data stored in non-XML formats may be integrated into the system. It is presented here since CAD and GIS support is fundamental for a system targeted at the construction industry and to present the general principles for handling non-XLM data. Additional examples of software that transforms information in proprietary formats to and from XML are presented within the chapter, Specified Mobile Application Server Design.

The GIS and CAD features are accomplished by using Open GIS compliant server software, which is available from several leading vendors as Oracle and Autodesk. There are also open-source alternatives that support the Open GIS standards. The exact functionality of the server differs from implementation to implementation, but the basic functionality is the same. The server is capable of interpreting several common spatial file formats and store the data in a database with spatial support. Further, the server is able to perform some set of operations on the data and deliver

the result as GML, which is an XML standard for describing geographic data. The GML may then be transformed, as described in the previous section, to an XML based presentation format such as SVG (Scalable Vector Graphics).

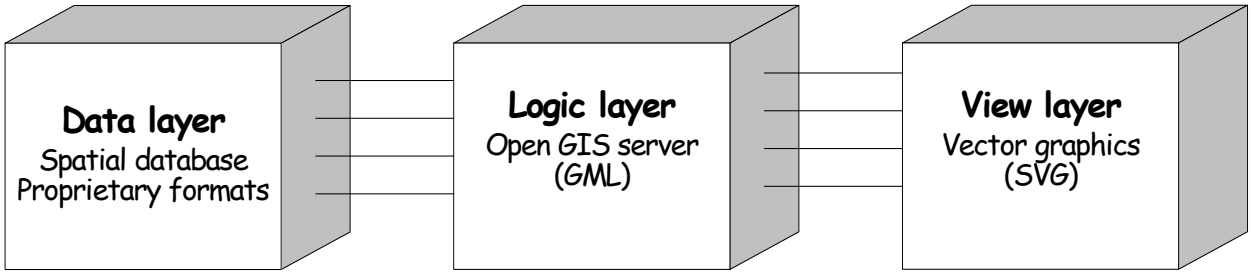



**Figure 5.4 GIS to SVG transformation**

The figure above illustrates the flow of data from the data layer to the presentation layer. A flow as the one above is called a pipeline. The pipelines are defined at the server but invoked by the clients. When a client requests the vector map the server first queries the GIS server, which fetches the spatial data, performs the required operations and outputs GML data. The GML is then transformed to SVG, using the stylesheet defined for the pipeline, and sent to the client. The pipeline is somewhat simplified to efficiently communicate the principles behind the design, a real world pipeline is defined in greater detail.

### 5.3.4 Handling Complexity

Pipelines have a tendency to become complex rather quick. If a client request a business report from the server it may trigger lots of actions. For example, charts and diagrams might need to be generated which triggers requests to a service that produces graphics. This service needs some data to generate the diagrams from, which it request from some other services like an ERP system or the GIS system described above. These systems in turn makes request to other services and so on. If one should define all transformations, aggregations etc for each new client request to support it would soon be quite complex; also it would require too much time and effort. The solution is to make a pipeline for each task and then connect the different pipelines to each other in a hierarchal fashion, as described below.

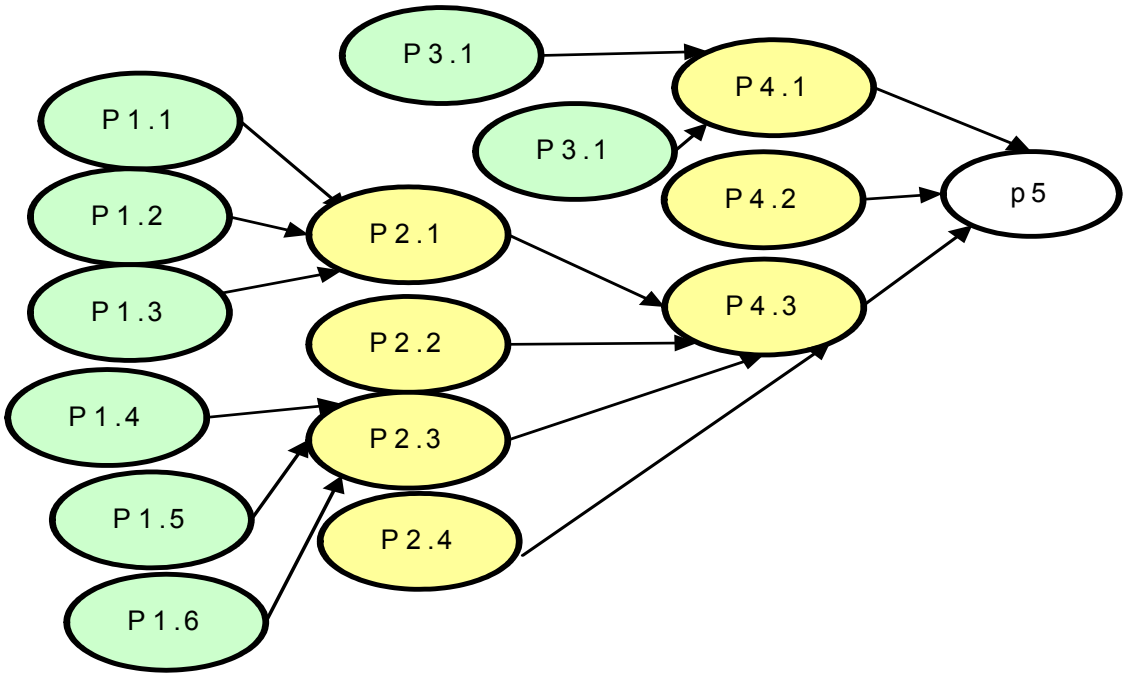



**Figure 5.5 XML Pipelines**

The connected pipelines can then be defined as single pipelines to simplify use. In practice one uses the pipelines as building blocks, building more and more complex information process from the processes already defined.

## *5.4 Data Management*

The type of data repository one should use depends on the structure of the data and how the data is intended to be used. One of the most common ways to store data is within a relational Database Management System (DBMS). A relational DBMS stores data as tuples in flat tables, which is good practice when the data is highly structured. This is generally the case when using an object-oriented design approach. A relational DBMS also works well when one needs to store and access documents on a per document basis (Fiebig et al, 2002). The problem with the relational DBMS is that it does not perform well when one needs to get parts of a document instead of working with it as whole, which often is the case when working with XML.

### 5.4.1 Relational Database Management Systems

The EJB container uses a relational DBMS to store EJBs and other information that needs to be stored. The GIS server also uses the relational DBMS to store spatial data. If the server is extended with additional functionality in the same fashion that the GIS and CAD functionality was added, those services will probably also use the

relational DBMS as their data depository. In short, the relational DBMS is the standard way of storage in the presented design.

### 5.4.2 Native XML Databases

The presented design relies heavily on the use of XML and the possibility to make queries to retrieve parts of XML documents. As mentioned above this is an area where a traditional relational DBMS performs poorly. An alternative approach, which is becoming increasingly popular, is to use a native XML database.

The term Native XML Database (NXD) is deceiving in many ways. In fact many NXDs are not really standalone databases and do not store the XML in true native form. Below is an attempt to define what makes a NXD.

A native XML database defines a logical model for an XML document, as opposed to the data in that document, and stores and retrieves documents according to that model. At a minimum, the model must include elements, attributes, PCDATA, and document order. Examples of such models are the XPath data model, the XML Infoset, and the models implied by the DOM and the events in SAX 1.0.

The NXD has an XML document as its fundamental unit of logical storage, just as a relational database has a row in a table as its fundamental unit of logical storage. It is not required to have any particular underlying physical storage model. For example, it can be built on a relational, hierarchical, or object-oriented database, or use a proprietary storage format such as indexed, compressed files.

This makes a database that is specialised for storing XML data, which stores all components of the XML model intact. Documents go in to and documents come out of the NXD, which may not actually be a standalone database at all.

NXDs are not a new low-level database model, and are not intended to replace existing databases. They should instead be considered a tool intended to assist the developer by providing robust storage and manipulation of XML documents.

As already stated, much of the data that is supposed to pass through the *Mobile Application Server* will be in some XML form. It is irrational and time consuming to parse all this data and map it to an Object-Relational model. Therefore the *Mobile Application Server's* relational DBMS will be complemented with a NXD.

Below is a figure illustrating the shift in application infrastructure, which is a result of the proper use of an NXD. By using a NXD the need to convert and from XML may be eliminated or greatly reduced.

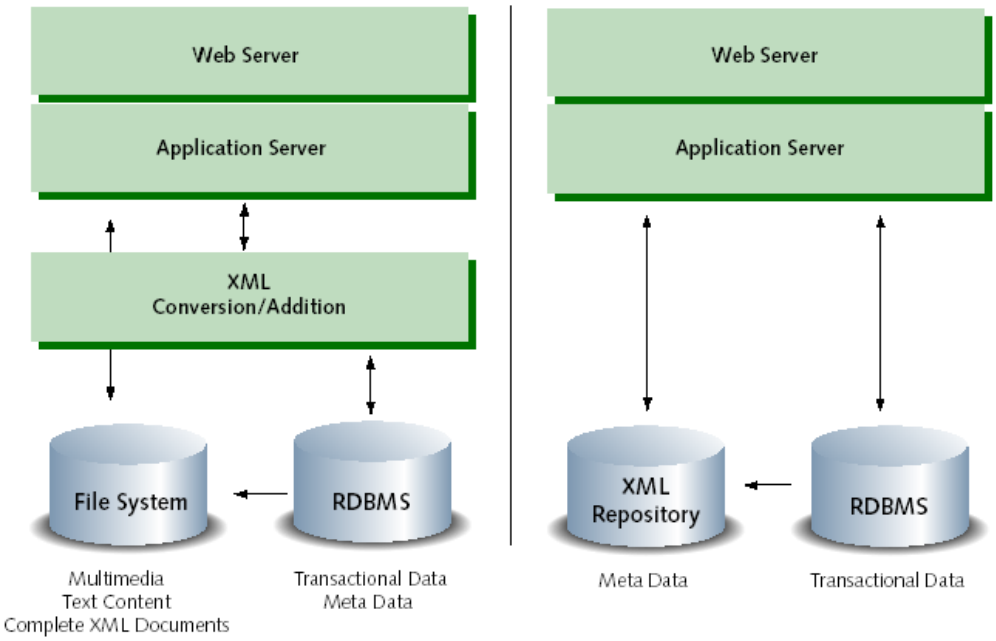



**Figure 5.6 Handling XML data, ZapThink**

## *5.5 Enterprise Information Portals*

When designing an enterprise information system there is often a desire to offer a single aggregated, personalised and customisable view of the system for each user. It is also a common that one wants to incorporate some information from outside the system like stock quotes and news headlines in these views. Another feature that is frequently sought after is the possibility to accomplish a consistent interface for all services, so the user always feels at home irrespective of which service is currently used. Enterprise portal software is the type of software used to meet requirements as these. There are many different vendors of enterprise portal software and there are also open-source alternatives available. The design presented within this chapter to some extent relies on the fact that there is portal software present, which can benefit from the features offered. If the enterprise information portal is seen as a part of the design or not depends on which view of the design is applied. Below is an overview of the different features that normally constipate a portal system.

Content Aggregation and Integration: One important aspect of a portal is its ability to integrate applications, web services, and content. This functionality includes the ability to embed non-persistent information, such as stock quotes, through the portal, and to run applications within, or deliver them through, a portal.

Search and Categorization: Search and Categorization are most often used when accessing a large library of documents. Search engines allow for the retrieval of documents based on criteria specified by the end user. Categorization allows for the organization of these documents into a browse-able hierarchy.

Content Management: Content management functionality is critical to managing all of the content in the Portal. Content management covers the full life cycle of publishing information to the Portal ranging from content creation and collaboration to workflow, version control, staging, and publishing of new and updated content to the Portal. While content is most often internal data that is useful to the users of the Portal, it can also be external news feeds and data sources.

Security: Security is a key functionality in Portals. Security can address many different needs within the Portal, including authentication into the Portal, encryption of the communications between the Portal and the end user, and authorization of the content and applications to only those users that are allowed access.

Directory: Most Portals include a directory of some type as a centralized datastore. The primary purpose of this directory store is providing a single location for the storage of information concerning the Portal and it's users. Such information may include the personalisation settings of an end user, end user authentication information, and listings of the applications an end user is allowed to access.

Web Site Analysis: Portals can track the user activities. Business drivers for doing this include: tracking what services users are most interested in, and utilising data collected on users previous visits to further customise and personalise the Portal to meet user interests and needs. Analysis software is also important in keeping track of trends within a Portal: which services are being most used, and by which groups of users.

## *5.6 Mobility Aspects*

Mobile data management is an established research area. The asynchronous, message-oriented communication paradigm that is implied by the use of XML might turn out to be particularly suitable for mobile applications. Also, as resources in mobile computing will remain inherently scarce, personalisation and localisation of information, requiring adequate management of the corresponding metadata, will play a major role (Aberer, 2001).

### 5.6.1 Mobile Usage

The fact that the system is intended to be used in a mobile setting puts additional requirements on the system. For instance, the design needs to support telemetric services, synchronisation and standard formats as WML and SVG that is frequently used within the mobile domain. Also, one should consider measures to reduce traffic and enhance the usability of mobile services. These measures may be installing proxies that pre-process the data to ease the performance requirements on the mobile devices or enhanced client caching to reduce traffic.

An aspect that not solely relies upon the mobility aspects, but maybe more on the will to have loose connections between the different parts of the presented design and its surrounding, is the objective to expose business objects and tasks as web services. This leverages not only new mobile opportunities but also the possibility to integrate with other systems using a loose integration approach.

The presented design supports thin as well as fat clients, which is the two standard types of clients today. The thin client approach is bandwidth intensive, comparatively slow and page based. The fat client approach requires distribution of relatively large software clients that need to be updated continually. If a good framework is not in place to handle this type of issues the fat client approach may become an administrative nightmare with lots of different legacy clients to support. Today their exist no common adapted standard for this type of mobility framework. But, several vendors offer non-standard solutions that offer the functionality requested above. These software solutions work well with the design presented in this thesis.

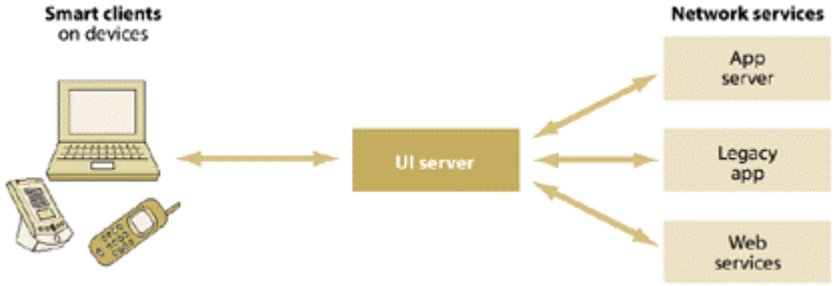



**Figure 5.7 An alternative design using a UI server, Gartner Group**

Some start-ups have tried to launch alternatives to the thin and the fat clients, which leverages the benefits with both approaches. One such approach is the use of a User Interface server, which makes it possible to have very little code on the actual client but still get the benefits of a fat client solution.

### 5.6.2 Telemetrics

Telemetrics is a technology that involves the automatic measurement and transmission of data from remote sources. The process of measuring data at the source and transmitting it automatically is called telemetry. The two terms, telemetry and telemetrics, are often used interchangeably.

In 1912, the first telemetrics application in Chicago used telephone lines to transmit operational data from a power plant to a central office. Because telemetry was originally used in projects like this, the first telemetry systems were called supervisory systems. In 1960, the interrogation-reply principle was developed, which allowed a more selective transmission of data upon request (Whatis, 2002).

Modern-day telemetrics frequently uses wireless communication. Telemetrics applications include measuring and transmitting data from space flights, meteorological events, wildlife tracking, camera control robotics, and oceanography studies. The Global Positioning System (GPS) is also considered to be a telemetric technology.

In a construction industry context there are several potential uses of telemetrics, one is as a tool to monitor changes in the environment during larger construction projects. Another way to benefit from telemetrics is to monitor vehicles and machinery. By measuring for example the amount of vibrations in a piece of machinery it is possible to identify a potential breakdown before it occurs and causes a standstill.

For a *Mobile Application Server* designed for the construction industry it is natural to support not only human users but also the vehicles and the machines that are usually used at a construction site. Despite this, the functionality to handle the physical connection of machines is not within the scope of our system. Hence a system that handles this functionality must easily be integrated and care must be taken regarding how the collected data should be handled to accomplish maximum benefit.

An example of how a telemetry system may be integrated is presented in the prototype chapter. The integration makes it possible to use the server's features to produce comprehensive statistics on machine use and status to assist evaluation and planning tasks. It also makes it easy to implement functions to alert the appropriate personnel when a machine needs maintenance or a limit value has been exceeded. The ERP systems integration opportunity facilitates automation of invoicing routines. Finally the fact that usage data are logged in real-time makes it easier to settle potential disputes later on.

In the long run telemetrics may change how business in general is conducted, but maybe not as much in construction as in other industries. New business models can evolve benefiting from the possibility to charge for service instead of time. Also better customer care and new service offers should be possible with diversified service and warranty packages. In the short perspective telemetrics may be used to increase efficiency, reduce risks and to produce information to make more informed decisions.

### 5.6.3 Synchronisation

A system designed for use in a mobile environment must be able to handle a loss of the communication link between client and server, such a loss may occur unintentionally as well as intentionally (i.e. the link is malfunctioning or the user makes the choice to disconnect). Systems with clients that cease to operate when they

loose their connection to the server require extremely reliable data communication links. To put such a restraint on a system limits its possible area of use significantly.

If a client should be able to function also during periods of server unavailability some data needs to be stored on the client. This raises a number of questions, like which data should be stored and how will the consistency of the data be preserved. To handle consistency the server's data and the client's data needs to be synchronised after a loss of the data communication link. With regard to which data to store on the client, we have decided that when using a rich client it's appropriate to store vital system data and a cache with the most recently accessed data. The size of the cache should depend on the client and be implemented using a first in first out approach. In many cases such a solution makes it possible for a user to continue working with the task at hand without interruption or data loss, despite server unavailability. Fat clients may use a light database on the device and more complex synchronisation rules.

To perform the synchronisation an appropriate communication protocol is required. A synchronisation protocol should define the workflow for communication during a data synchronisation session when the mobile device is connected to the network. The protocol must support naming and identification of records, common protocol commands to synchronise local and network data, and support identification and resolution of synchronisation conflicts.

SyncML is an XML-based language for synchronizing devices and applications over any network and provides such a protocol. With SyncML, networked information can be synchronized with a mobile device, and mobile information can be synchronized with a networked application. With SyncML personal information, such as email, calendars, to-do lists, contact information and other relevant data, will be consistent, accessible and up to date, no matter where the information is stored (SyncML, 2002).

SyncML minimizes the use of bandwidth and can deal with special challenges of wireless synchronization like relatively low reliability of the connection and high network latency. A particular strength with the SyncML standard is the broad industry support. Large mobile device manufacturers like Nokia, Motorola, Palm and Ericsson have support for SyncML in their more recent models. Also large enterprise software companies like IBM and Lotus support the standard in their products. SyncML provides our design with a proven reliable synchronisation framework. The wide adoption of the standard also makes it possible for our system to synchronize standard information such as calendars, to-do lists and contact information with other enterprise information systems and a large number of mobile devices. But this solution has some performance issues because of the need to translate and process XML for each request. The solution should therefore be complemented with the possibility for a rich client to use a more efficient protocol like RMI-IIOP, letting the client speak directly to the EJB layer (Kao, 2001).

# 6 Specified Mobile Application Server Design

While the previous chapter described the general design this chapter gives examples of specific software that may be used to implement the proposed design. Both open-source and commercial software alternatives are introduced for each part of the server. Finally a complete suite of open-source software that may be used to implement the design is presented along with some examples from our own implementation.

## *6.1 Software Components and Frameworks*

This section introduces some examples of software that may be used to implement the proposed design. Some guidelines are also presented with regard to on which basis the choice of components should be made. It does not describe the functionality of the different components, which is described in the previous chapter.

### 6.1.1 EJB Containers

The core in an J2EE based application server is the EJB container and its related components. Since the J2EE specifications are continuously changed and improved, the vendors' ability to quickly adopt the new specifications is an important competitive factor. But there are several other factors that need to be considered when choosing an application server. Which factors that is most important depends on the needs of the client. For a large implementation factors as scalability and clustering is of outmost importance. For a web based application the performance of the web container is often a limiting factor. The web container is not a part of the EJB container but the two are usually sold together as an integrated package. This package also includes other complementary software that it is useful when developing enterprise applications. Exactly which software is included depends on the vendor; different vendors have chosen to target different customers.

BEA Systems is the dominant actor among large companies. BEA's server, Weblogic, is very scalable and BEA is always quick to implement new specifications. BEA also offers a comprehensive line of complementary software like integration frameworks. The biggest drawback with Weblogic is the price of the product. BEA's toughest competitor is currently IBM, which with their server Websphere offers a similar business offer. BEA is a company that is focused exclusively on application servers and related products. IBM on the other hand offers a broad range of services with a focus on e-business infrastructure. BEA is generally regarded as having a better application server than IBM and when making a best of breed solution Weblogic is often chosen as preferred server. IBM on the other hand has the ability to offer a complete e-business solution, as a result customers that want to buy an entire solution from one single vendor often choose Websphere as application server.

Another actor that trying to offer a complete e-business suite is Oracle, which is becoming more and more of a serious contender in the application server marketplace. Oracle started to strengthen its market position in 2001 after licensing an EJB container from a small Swedish company called Ironflare AB. Ironflare's application server, Orion, is one of the fastest if not the fastest of the different J2EE application servers that are available today. Oracle's application server Oracle9iAS is an interesting alternative to the leading two since it has good performance characteristics, a comprehensive line of complementary products and a lower price than Weblogic and Websphere. In numbers of sold licences on a yearly basis Oracle has quickly reached a level that is comparable to IBM and BEA, but in revenues they are still significantly behind the leading two.

The last mayor player in the J2EE application server field is Sun with their server Sun ONE. They have lost ground lately but have a relatively strong position partly because of their special relation to Java technology. There are also several niche actors, which are strong within certain market segments.

Due to Sun's licensing terms there are no certified open-source J2EE application servers. But there are open-source servers available that comply with the specifications; the most prominent one is JBoss. JBoss is one of the most advanced application servers on the market and the JBoss team is very quick to implement additions to the specifications. The server differs from the commercial ones not only by being open-source but also by only being an application server and nothing else. There are no suits of complementary products bundled; the focus is on making the best application server and nothing else.

The trend is that the EJB container becomes more and more a commodity and that customers demands a complete integrated suite of products that meets their needs. This trend is supported by the success of JBoss and the fact that some commercial vendors like Hewlett-Packard offers their EJB container free of charge.

All of the more popular J2EE application servers may be used to implement the design, that is one of the basic thoughts behind the design. No vendor specific functionality is used in the design so no such considerations need to be made. But some rather new features as Container Managed Persistence are encouraged by the design and all vendors have not yet implemented proper support for this.

### 6.1.2 Web Containers

Most commercial application servers have an integrated web container to handle servlets and related technologies like Java Server Pages. Among the leading commercial application servers Oracle has one of the best performing web containers. There are also vendors that have primarily focused on delivering good performance with regard to JSP and servlets. Such vendors are Macromedia with their JRun server and Caucho Technology with their server Resin. If an

implementation is primarily JSP and servlets based these options needs thorough investigation.

The two main open-source alternatives are Tomcat and Jetty. Tomcat from Apache is Sun's official reference implementation and a well-regarded web container. Tomcat has had some performance issues in the past but the new Tomcat container, Catalina, has been redesigned from the ground up to resolve this. Jetty is small, efficient and fast and therefore popular to integrate in other products like the JBoss application server, which uses Jetty as the server's default web container.

### 6.1.3 XML Frameworks

Some sort of XML framework is included in most major application servers. The XML functionality offered is one of the more important add-ons the different vendors offer. Since there is no clear definition on what makes an XML framework and because the functionality differs considerably between vendors no comparison is presented. The major open-source alternative with regard to XML Publishing Frameworks is Cocoon 2 from the Apache Software Foundation. Since it is used to such a great extent within the presented design it is described in greater detail below.

Apache Cocoon is an XML publishing framework developed on top of the Apache Avalon Server Framework. It allows a developer to define XML documents and transformations to be applied on it, to eventually generate a presentation format of choice (HTML, PDF, SVG, etc.). Cocoon also gives the developer the possibility to have logic in XML files so that the XML file itself can become dynamically generated.

Cocoon uses a design model as the one described in the figure below, which outlines four major concern areas and five contracts between them. Cocoon is engineered to provide a way to isolate these four concern areas using just those 5 contracts, removing the contract between style and logic that has been bugging web site development since the beginning of the web.

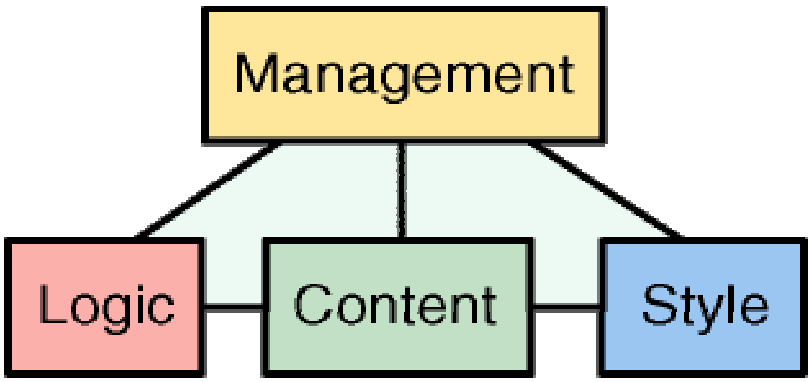



**Figure 6.1 Separation of Concerns, The Apache Cocoon Project**

Cocoon is designed to allow Developers, Business Analysts, Designers, and Administrators to work with each other without breaking the other person's contribution. The problem with only using JSPs, ASPs, or ColdFusion templates is that all of the look, feel, and logic are intertwined. That means that maintenance is much more difficult, and the project's true costs are delayed until the customer wants feature enhancements or bugs fixed.

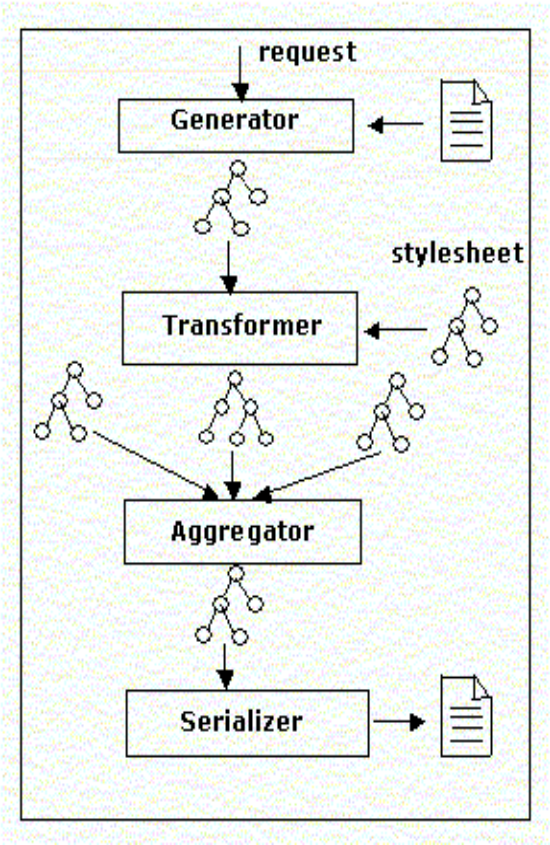
Developer's jobs are to create the business logic and object model behind the web application. They are more concerned with functionality than with layout or the words displayed on a screen. These are the people that develop the Actions and the hooks for how to get the necessary information from business objects. An Action is a component that only process information.

The Business Analysts are the people who are concerned with the words displayed on the screen, and to a certain extent, the layout. Typically, they will be using the work done by the developer to put together a generic markup that will be transformed into the results. In small development environments, many times the developer takes on both this role and the developer role. Typically, the business analyst works with the markup language that goes into the generator.

The designer is the person or group of people who are responsible to provide the final look and feel of a site. The designer does all the graphics and HTML code. In Cocoon, they work with the Transformers that take an input and structure it in a final presentation.

The administrator is responsible for the sitemap that maps the URI space to the different pipelines in Cocoon. A pipeline is a path from a Generator to a Serializer. This means that the administrator decides that all requests for a resource with an ".html" extension starts out as XML and ends up as HTML. The Administrator works closely with the Designers and the Developers. In the absence of a dedicated administrator, one developer should assume that role. It is important that developers do not get bogged down in this one Component.

The way Cocoon handles a request is powerful but may be a bit hard to grasp at first. Below is a brief overview of the route from request to response. For a more thorough overview see the Cocoon website at http://xml.apache.org/cocoon.




First the incoming request is tested against the match patterns in the sitemap and the request is dispatched accordingly.

Secondly the XML documents are generated (from content, logic, Relational DB, objects or any combination thereof) through Generators.

Then the XML documents are transformed through Transformers (to another XML, objects or any combination).

The XML documents are aggregated through Aggregators.

Finally the XML response is rendered through Serializers.

**Figure 6.2 XML Transformations, The Apache Cocoon Project.**

### 6.1.4 Relational Databases

According to a Gartner Dataquest market analysis from may 2002, there are two dominant vendors of relational databases, IBM and Oracle with about one third of the market each. Oracle had a slightly larger share of the market until recently when IBM passed by acquiring Informix, another leading database vendor. Microsoft is third in the market with more than 16 % market share; they differ from the other leading actors by showing significant growth. The fourth of the larger database vendors is Sybase, which is losing market shares but still retain roughly 3 % of the market. The relational database market has matured and all the remaining products from the larger database vendor are all high quality products. Which database to choose is a matter of personal taste and in some cases a need for some special feature offered by a particular vendor. Such a feature may be the ability to easily replicate data from a lightweight version of the database. Both Oracle and Sybase offers lightweight databases with a small footprint designed for wireless devices and other devices with limited resources.

There are several opens source databases available, but two of them have a particular strong standing within the open-source community. The two are PostgreSQL and MySQL and they are both reliable and fast enough to be considered when implementing enterprise systems. There are also free open-source databases

available from commercial vendors. The German ERP-systems giant SAP is one of the companies that offer a database free of charge.

### 6.1.5 Native XML Databases

The native XML database market is still in an early adopters face. Tamino from Software AG dominates the market with 48 % of the market. eXcelon is number two in the market with a 25 % market share and they differ from other vendors by offering an XML-Object database instead of a pure NXD. Other NXD vendors include X-Hive, NeoCore, IXIASOFT, Ipedo, XYZFind, PyBis and BirdStep. Behind the two leading vendors there is a very close race with several actors with about the same market penetration. In contradiction to the situation in the mature RDBMS market there is still room for new actors in the NXD-market. The market experiencing significant growth, the NXD-market grew 400 % in 2000 and 189% in 2001. The expected growth for 2002 and 2003 are 230 % and 250 % respectively, according to a recent market analysis by ZapThink.

There are two major open-source alternatives Xindice and Ozone. Xindice was previously known as dbXML but changed its name when the source had been donated to the Apache Software Foundation. Xindice is a pure native XML database while Ozone is an XML-Object database. Ozone has many interesting features that are not XML related like the ability to handle persistent Java objects so they may be coded as their were ordinary Java objects.

### 6.1.6 Portal Software

Most major application server vendors offer portal software as an add-on, there are also portal software companies specialising in enterprise portals. One such company is Plumtree, which is the market leader with in the field of corporate portals. When deciding which portal software to use the choice is generally between the application server vendors offer and offers from companies like Plumtree specialising in portals. Using software from different application server vendors together in a truly integrated fashion is not common. But of course it is not unusual that application servers from different vendors coexist and cooperate within a company's IT infrastructure.

Currently Jetspeed from the Apache Software Foundation is used as portal solution in our implementation. Another Apache project, Cocoon, has developed a portal solution for the Cocoon framework but it is still in a beta stage. When the Cocoon projects solution matures it will probably be a more feasible solution for a design as the one presented in this thesis. The Cocoon solution is favourable since it to a larger extent benefits from the possibilities offered by the XML publishing framework that supplies the information.

## 6.2 Our Implementation

Our implementation of the general design is completely based on open-source software. The design has evolved during the course of the thesis project and changes to the specified design have been done on a regular basis. For example, initially the Orion application server was used for its good performance and ease of setup. But later on a switch was made to JBoss to offer a solution solely based on open-source software.

The only part of the design that can be regarded as missing in the implementation is the synchronisation framework; this is due to the fact that we have not find an suitable open-source alternative. Below is an architectural overview of our *Mobile Application Server* implementation, compare it to figure 5.2 on page 31 to see how the implementation corresponds to the general design.

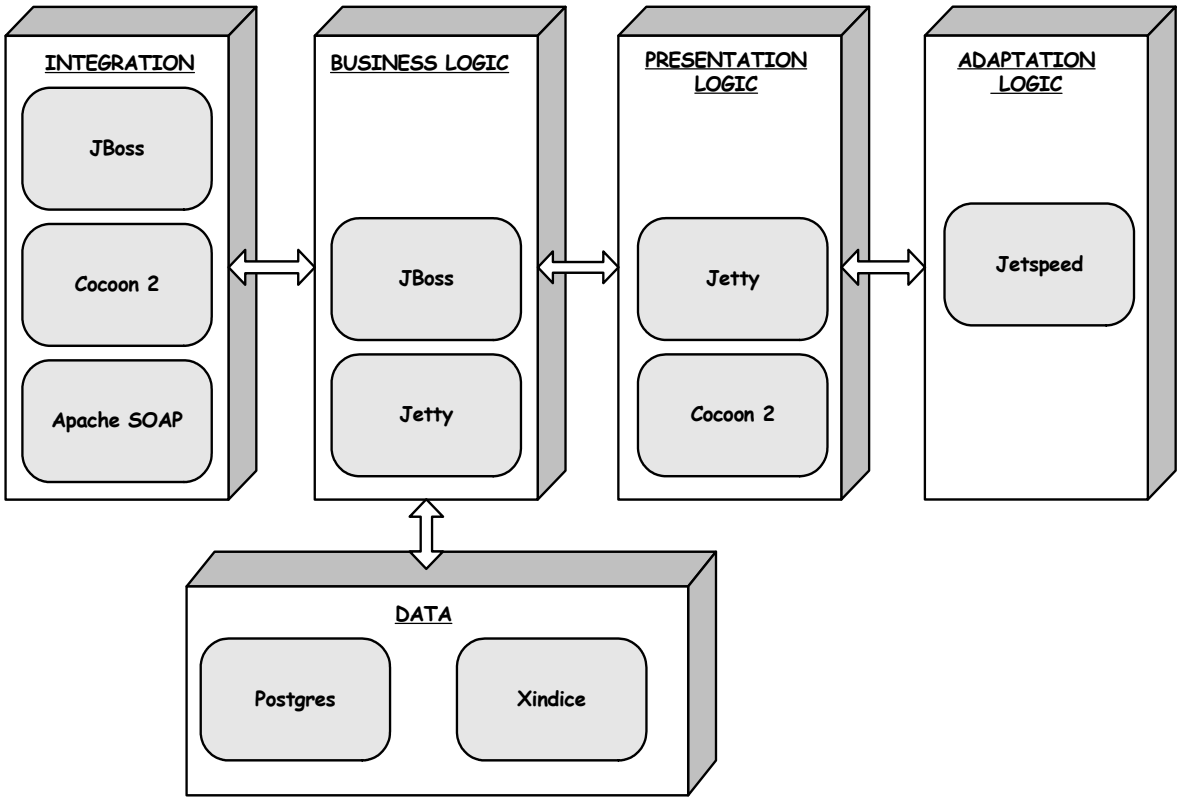



**Figure 6.3 Specified Server Architecture**

To test the *Mobile Application Server* platform some supplementary software in addition to the core components featured above has been installed on the prototype server. The software in question is a GIS/CAD server called GeoServer and a PostgreSQL extension called PostGis, which makes it possible to store geographical objects directly in the database. Also a telematics platform called *Aptus* from *Pilotfish* has been integrated with the *Mobile Application Server* using XML-messaging.

### 6.2.1 Core Components and Frameworks

All the components and frameworks selected for our implementation has been mentioned earlier in this text. The software used is open-source; this is due to several reasons. One reason is to show that it is possible to build a complete feasible solution with only open-source software. Another reason is that open-source projects generally comply with open standards, which is important aspect in this design. There is also a practical aspect, even though most software vendors within this niche offers trial downloads, the open-source software is the most accessible.

As an EJB Container we have chosen JBoss because it is a free and very stable container. Another important factor for choosing JBoss was the JBoss Group ability to quickly implement new standards. Our choice of Web Container was partly guided by our choice of EJB Container; the decition was made to go with Jetty for performance. When choosing which XML Publishing Framework to use Cocoon 2 was our only alternative. Cocoon is the only comprehensive XML Publishing Framework we have found that not rely on proprietary technology. Also when compared to proprietary solutions Cocoon appear to be a platform to be reckon with.

Our current implementation has only limited support for Web Services in the form of Apache SOAP; the problem is that different SOAP implementations are not completely compatible.

As the relational database PostgreSQL has been chosen, it is in fact an object-relational hybrid database but it performs the same tasks as the relational database outlined in the general design. PostgreSQL was chosen since it is very reliable database and because it is easy to create additional data types for the database. This ability was used when additional support for spatial objects was needed for a GIS prototype developed within in the scope of this thesis.

Xindice from the Apache Software foundation has been chosen as native XML database since it is the leading open-source NXD. Ozone is another of the more prominent open-source databases. It is an XML-Object database and it was considered for the implementation but in the end the choice was made to go with a pure NXD in compliance with the general design.

Jetspeed from the Apache Software Foundation is used to provide the portal functionality in our implementation. Several of the components and frameworks used in the implementation is Apache software. This is due to the fact that the Apache Software Foundation has a strong developer community and a reputation of offering commercial quality software. Also the different Apache groups collaborate to a large extent so software from different groups often works very well together. Further the different Apache projects frequently incorporate software from other Apache projects in their solutions. All this has made it possible for the Apache Software Foundation to offer a rather comprehensive line of different software packages that complement each other.

### 6.2.2 Sample Client

In order to try out the platform some rich clients for the Compaq iPAQ was constructed. Two different iPAQs were used one 3630 with 32 MB of memory and a 3660 with 64 MB of memory. Both PDAs were equipped with a WLAN adapter and run the SavaJe OS from SavaJe Technologies. The SavaJe OS made it possible to use Java 2 Standard Edition when developing the prototypes, which eased the development work significantly. Normally, Personal Java or Java Micro Edition is used instead when developing Java applications for PDAs due to resource limitations. Below is a screenshot from a simple web browser for the iPAQ, made with Java's standard classes.


**Figure 6.4 Java Web Browser**

### 6.2.3 GIS Functionality

The OpenGIS Consortium is an international industry consortium of more than 220 companies, government agencies and universities participating in a consensus process to develop publicly available geoprocessing specifications. Open interfaces and protocols defined by OpenGIS Specifications support interoperable solutions that "geo-enable" the Web, wireless and location-based services, and mainstream IT, and empower technology developers to make complex spatial information and services accessible and useful.

The GeoServer project offers a free open-source J2EE implementation of the OpenGIS consortium's Web Feature Server specification. The choice was made to use this server in order to support GIS and CAD functionality in our implementation. See section 5.2.3 and figure 5.4 for a general view of the flow from data layer to logic layer to the final presentation at the client. In that example the logic layer just transformed the data from a native format to GML but the process may be a lot more complex.

The GeoServer is designed to act as a thin, portable, OpenGIS-compliant web service layer on top of existing data sources. When all data is transformed into GML it is possible to make calculations and transformations with standard tools. This might be route optimising or calculating the cost of different transportation alternatives using the GIS-data and business data from ERP systems.

When there is a need to display the GIS-data in the form of a map or other graphical representation SVG is the natural choice for a design as the one presented in this thesis. SVG is the W3C standard for scalable vector graphics and an XML-based file format. The fact that both GML and SVG are XML-based makes it possible to use standard XML tools, as described in section 5.3.4, when transforming the GIS-data to a viewable image.

In our implementation the Batik SVG Toolkit from the Apache Software Foundation handles the visual presentation of GIS data. The toolkit comes bundled with Cocoon but it may also be used on its own, as is the case with the sample client presented in 6.2.4. The toolkit is capable of three basic things generation, manipulation and viewing of SVG images. Generating means the ability to generate SVG from non-XML sources as bitmap images. Manipulation is the functionality to make complex transformations to the SVG images. The viewing functionality is accomplished by specially designed Swing components that may be used to display SVG images within a Java application.

The power of our way of implementing the GIS functionality besides the functionality offered is modularity and vendor independence. Most major spatial databases may be used as data repositories; the same is true for the major map server vendors. Cocoon and Batik may also be exchanged for other frameworks. The same

goals that guided the development of the general design should guide how additional functionality as this is added to the *Mobile Application Server* solution.

### 6.2.4 Telematics Platform

The purpose of a telematics platform is to make it easy to monitor and control machines remotely. For our implementation we have chosen the Aptus platform from the Swedish telematics company Pilotfish. Aptus consist of two basic parts a server with a database and wireless gateway units that are mounted on the machines that are to be connected. The machine functions that need to be monitored or controlled are connected to the wireless gateway unit. The different gateway units then communicate with the server over a wireless carrier. The server stores the collected data in a database and it is also at the server that all settings are made, as to when to collect data etc.

Our server interacts with the Aptus server using XML messaging. No vehicles or machines have been connected for this project; instead an already connected commuter vessel in the Gothenburg harbour has been used to try out the platform. Below is a sample screenshot from a small test application showing an SVG-map by the help of the Batik SVG Toolkit.


**Figure 6.5 Sample SVG Client**

# 7 Discussion and Conclusion

This chapter discusses the thesis results in the form of our conceptual design and our specified design. In the first section we discuss the construction specific features of the design an why it is of relevance to the construction industry. In the second section we present the different characteristics of the proposed system design. The characteristics include performance issues, system scalability and flexibility, system maintenance, vendor dependency and hardware options. In the last section we discuss matters regarding a future system implementation within an organisation working in the construction industry.

## *7.1 Supporting the Construction Industry*

Depending on the perspective the proposed design could obviously be applied on other problems than the ones presented in the introduction chapter. The use of mobile computing is evolving in many areas. General software applications such as packages for project management, planning and scheduling do not differ very much when for instance comparing the automotive industry with the construction industry. However when working on a construction site there is no permanent infrastructure and the infrastructure used in most construction projects is supplied via mobile equipment ranging from diesel generators to office premises. Wireless networks and the use of mobile devices to accesses existing information systems could solve a number of the problems presented earlier. The arguments for using mobile computing and structured information in the construction industry are documented by several authors (Magdic et al, 2002), (Cox et al, 2002).

When distinguishing the core construction industry features of the proposed design we would like stress the utility to display technical file formats such as CAD and GIS that constitute a very large base of the actual information handled on a construction site. By applying the components included in the design it is possible to view and edit for instance a drawing made in Autodesks AutoCAD package on a Compaq iPAQ handheld computer. The actual access to technical file formats on a mobile device has been one of the primary goals during the work with the system design. In order to realise the above mention functionality a number design issues of more general nature obviously have to be addressed and analysed.

## *7.2 System Characteristics*

The specified design is to a very large extent based upon java based software components. As java code never is compiled into machine instructions, but interpreted by a java virtual machine there are obviously faster solutions. With the hardware capacity of to day, and certainly the next generation of computers the execution speed is not the constraining factor. Since the actual tasks performed from a mobile client do not involve any hard real time constraints, we argue that the proposed java-based design is a plausible solution. The main purpose of choosing java-based software components is to obtain platform independence. Platform independence makes it possible to design and build the system without taking into

account platform dependent constraints. The proposed design makes it possible to easy migrate from one system to another and this notion could be very useful if the migration involves an actual change of operating system.

By maintaining a high level of flexibility and minimising coupling between software components and software frameworks within the system, there will be performance losses. One objective in the system design is to ensure database independence and this implies that the database communication is not fully optimised. The design recommends the use of third party frameworks to automate the optimisation of database queries. The result is a design where the database communication seldom is fully optimised, but always relatively fast.

Databases Management Systems are traditionally often the base of information systems and very often they are strongly integrated in the overall information system solution. Our solution takes mobile information access as the starting point for a system design and the flexibility to easy exchange the database is central the design.

Our solution proposes the use of both a relational database and a native XML database. These two databases have overlapping functionality but their feature sets differ considerably. Whereas relational databases are tuned for long term storage of durable data in the back end, XML databases have emerged as a practical add-on for managing active data between systems in the middle tier. By decoupling active data form durable data it is possible to simplify the development and ensuring flexibility.

The general concept applied for data presentation is the use of XML. Parsing XML-data in order to present the same information on different clients ensures client independence. When using thick clients for instance a PDA with a relatively powerful CPU, the production and parsing of XML on the actual server will cause losses in performance. Again the design choice favours flexibility and also client independence before performance. Since our design is constructed of two main technology approaches, namely Java 2 Enterprise Edition and the use of an XML-publishing framework, it is possible to extend the design with for instance client applications utilising the computing power of a thick client. We would also like to stress that the discussion regarding system performance is rather theoretical. While not utilising the full capacity of the hardware is not the same thing as a user finding the system performance not adequate.

By choosing J2EE and the different XML technologies, presented earlier in this thesis, as the fundamental building blocks of a mobile application server it possible to obtain the following characteristics:

1. The system supports the construction industry XML based information exchange standards bcXML (building and construction XML) and aecXML (Architecture, Engineering and Construction XML).

2. Vendor and platform independence, the J2EE framework makes it possible to choose among several software vendors and since java code is run in a virtual machine, the design is also platformindependent.

3. Concurrency, by using a J2EE compliant application server support for multi threading and transactions is enabled. Synchronisation is enabled via the use of SyncML.

4. Multiple mobile client support, the presented design supports the use of generic clients through the use of XML publishing frameworks.

5. Scalability, by making the system component based to a high degree ensures scalability and maintainability and the JEE framework is a proven technology in terms of scalability.

Returning to the initial research question and the sub questions that was derived from the initial question we mean that the above five characteristics together gives *one answer* the problem of designing a mobile application server targeted at the construction industry. The five characteristics clearly maps to the five sub questions presented in chapter one.

## *7.3 Implementation*

In order to make use of the *Mobile Application Server* a construction company or any organisation within the construction industry must have an already existing IT-infrastructure. The server itself is just gateway to existing information systems and it is not design to perform information processing in a self-contained manner. Since the design is based upon an application server supporting the Java Enterprise Edition frameworks it is possible to add data processing functionality to the design.

We believe that an organisation wanting to implement our design should have a rather high level of IT-maturity. Information automation is more common in larger organisations and we argue that our design would mainly interest these types of organisations. In any larger construction project there are a number of actors involved, such as customers, contractors, subcontractors and architects, and it is possible to use this system among these actors. One scenario could be that the major contractor owned the *Mobile Application Server* and that the subcontractors could access the system through various information portals optimised and of course restricted to their matter of interests. There are obviously a number of aspects to investigate in order to implement a system as described in our design. As stated earlier in this thesis we do not address any effects of a future implementation of the suggested system. The study of IT-use is a common research area within the field of informatics but this project has a strong design focus. Issues such as IT knowledge

among construction workers, information visualisation on small terminals, wireless computer network concepts among others, are all tightly interwoven into the subject at matter: *Designing a Mobile Application Server for the Construction Industry,* but they are not included in this study.

Companies spend a lot of money on software systems, and to get a return on that investment, the software must be usable for a number of years. The lifetime of software vary variably, but many large systems remain in use for more than 10 years. Some organisations still rely on software systems that are more than 20 years old. These types of system are often referred to as legacy systems. Re-engineering and integration of legacy systems in new software architectures have very much been one of the major tasks for software engineers and computer scientists, during the last ten years. The *Mobile Application Server* is designed to handle legacy system integration through standardised interfaces. One objective in using third party software components and software frameworks in our design is to make the system scalable and maintainable, and thus avoiding many of the common problems with legacy systems. The component-based architecture makes it easy to extend, modify or exchange components. In order to implement the system the idée is to perform a number of organisation dependent design and development decisions during the actual implementation and thereby producing a system that is optimised for the organisation in question.

Systems that require special design and development tasks are inherently more expensive than a standard solution. While it would not be sufficient to distribute our system as standard bundle it should be possible to reuse large portions of earlier designs to cut costs and development time.

In order to establish knowledge concerning standardised mobile system developed for the construction industry in general, further research must be performed. It is likely that for some groups within the construction industry existing mobile applications are fully satisfactory. If for instance viewing CAD-files and keeping track of basic project management information, on some kind of mobile terminal, are the major tasks of interest; there are several applications to choose from. Magdic et al (2002) performed a case study at a construction site, using only standard PDAs and standard stand-alone applications for the PDA. Even though the results were very positive, they concluded that the information systems in construction needs to be redesigned to attain a higher degree of information integration and a higher efficiency of information. A design as the one presented in this thesis may play an important role in accomplishing this.

# 8 Further Research

Undertaking a study like this is merely the beginning when designing mobile IT targeted at the construction industry. One possible way of continuing on the path would involve building a prototype system that would cover enough functionality in order to evaluate performance issues and the designs ease of use. Also there are some design issues that need further attention, both Web Services and Enterprise messaging is technologies that probably could be used to improve the presented design. Further there are plenty of technical questions to investigate, that we have not covered in this study since its focus was the actual design. The proposed research path could be projected using the conventional software engineering process model analysis, design, development etc. where the next stage would involve a strong development focus.

With a working prototype system research could continue with case studies at physical construction sites where people on the site would be equipped with mobile terminals. The case studies could than be analysed from a number of research perspectives including, application technology, project management, human computer interaction, IT education and various organisational aspects. We believe that mobile computing in construction do have great potential in solving many of the common problems when it comes to managing and making use of information.

# 9 Final Reflections

In order to utilise mobile computing within in the construction industry some kind of system, which acts as a bridge between the mobile devices and the existing information systems, is necessary. This thesis presents a design solution that theoretically could act as the gateway between the existing systems and the mobile terminals that would be used on a construction site. We argue that the proposed design could be used as starting point when embarking on the journey towards mobile computing in the construction industry. The design is not to be regarded as a final solution in any way but merely as the first leg of an ongoing iterative design and development cycle.

Our design is built around two types of technologies namely Java 2 Enterprise Edition and present and emerging XML technologies. By using these two types of technologies we have been able to present a design that is vendor and platform independent. The design is also able to support the current and upcoming information exchange standards of the construction industry. By applying design patterns and software engineering best practices we have been able to design a highly modular n-tier solution. These features make the design general and flexible enough to facilitate various modifications and extensions. When looking for alternative technologies we argue that there is really just on alternative to J2EE namely the Microsoft .NET platform. By using .NET, vendor and platform independence is not possible. When comparing .NET to J2EE the previous one is a less mature and untested in large scale systems. On the other hand the strengths of the .NET platform is XML and web services features. We regard the .NET platform not as a competitor to but as compliment to the design presented in this thesis.

# References


AecXML. (2002, October 22) *The Architecture, Engineering and Construction XML*, [www document]. URL http://www.aecxml.org/

Alur, D., Crupi, J., Malks, D. (2001) *Core J2EE Patterns. Best Practices and Design Strategies*. Prentice Hall, Upper Saddle River, New Jersey, USA.

BcXML (2002, October 22) *The Building construction XML*, [www document]. URL http://www.eConstruct.org

Björk, B.C. (1999) *Information Technology in Construction: Domain Definition*. International Journal of Computer Integrated Design and Construction, 1(1), pp. 3-16.

Celander, L. (2002, October 22), *STEP – En skonsam men effektiv introduktion,* [www document]. URL http://www.azelia.se/bocker.html

Clark, J (2002, October, 22), XSL *Transformations (XSLT) version 1.0, W3C Recommendation*, [www document]. URL http://www.w3.org/TR/xslt

Cox, S., Perdomo, J., Thabet, W. (2002), *Construction Field Data Inspection Using Pocket PC Technology*,
In proceedings of the International Council for Research and Innovation in Building and Construction CIB w78 conference.

Dahlbom, B. (1999) *The Idea of an Artificial Science*. In Dahlbom, B., Beckaman, S. & Nilsson, S.B., *Artifacts and Artificial Science*, pp. 2-15. Available from http://www.informatics.gu.se/~dahlbom/

Dahlbom, B. (1999) *The New Informatics*. In Ljungberg, F. (ed.), *Informatics in the next millenium*, pp. 15-35, Lund, Studentlitteratur. Originally published in Scandinavian Journal of Information Systems, 8(2), pp. 29-48, (1996).

Fiebig, T. et al (2002) *Anatomy of a Native XML Base Management System*, Technical Report of the University of Mannheim: 01/02

Finch, E. (2000) *Net Gain in Construction*, Butterworth-Heinemann, Oxford.

Flood, L.R. & Carson, E.R. (1993) *Dealing with complexity. An introduction to the Theory and Application of Systems Science*, Plenum Press, New York.

Halter S.L. & Munroe S.J. (2002) *Enterprise Java Performance*, Prentice Hall, Upper Saddle River, USA

Harris, F. & McCaffer, R. (2001) *Modern Construction Management*, Blackwell Science, Oxford.

Holmquist, L.E. (2000*) Breaking the Screen Barrier*. Doctoral Dissertation, Department of Informatics, Göteborg University, Sweden.

Intellisync (2002, October 22), [web document]. URL http://www.pumatech.com

K. Aberer (2001) *Advanced XML Processing*, ACM SIGMOD special issue on Advanced XML Data Processing (Vol 30, No 3)

Kao, J. (2001) *Developer's Guide to Building XML-based Web Services with the Java 2 Platform*, Enterprise Edition, Sun Microsystems

Koenig, K. (2001) *Construction Industry Software Overview*, The Project Manager, vol. 8 no. 2.

Kristoffersen, K. & Ljungberg, F. (1998) *Representing Modalities in Mobile Computing*, In Proceedings of Interactive Applications of Mobile Computing, (IMC 98). Rostock Germany.

Kristoffersen, K. & Ljungberg, F. (1999), *Mobile Use of IT*. In proceedings of IRIS22, Jyvaskyla, Finland.

Magdic, A., Rebolj, D., Cus-babic, N & Radosavljelic, M. (2002) *Mobile Computing in Construction*, In proceedings of the CIB w78 – distributed knowledge in building conference.

Mathiasen, L., Munk-Madsen, A., Nielsen, P.A. & Stage, J. (1999) *Objektorienterad analys och design*, Student Litteratur, Lund.

McLaughlin, B. (2001) *Java & XML*, O'Reilly, Cambridge

Monson-Haefel, R. (2001) *Enterprise JavaBeans, 3rd Edition*, O'Reilly, Cambridge

Newcomer, E. (2002) *Understanding Web Services*, Addison Wesley, Boston

Orfali, R. & Harkey, D. (1998) *Client/server Programming with Java and Corba.* John Wiley and Sons, New York.

Partsch, G., Specker, A. & Weber, M. (1998*) Personalized Multimedia System for Optimizing Civil Engineering Processes,* In proceedings of the of the 2nd European Conference on Product and Process Modelling, BRE, Watford, London UK (ECPPM-98).

Patel, R. & Davidsson, B. (1994) *Forskningsmetodikens grunder – Att planera och genomföra en undersökning*, Studentlitteratur, Lund.

Pouria, A., Froese, B. (2001), *Transaction and Implementation Standards in AEC/FM Industry*, In proceedings of the 2001 conference of the Canadian Society for Civil Engineers, Victoria, BC, Canada.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. & Carey T. (1994) *Human Computer Interaction*, Addison Wesley, Harlow, England.

Pressman, R. (1997), *Software Engineering a Practitioners Approach*, McGraw Hill, New York.

Rebolj, D. (2002, October 22), *Need for intelligence in complex systems*, [www document]. URL http://ai.fri.uni-lj.si/xaigor/articles/rebolj.doc

Sekaran, U. (1992) *Research Methods for Business, A Skill Building Approach,* John Wiley & Sons, New York.

Sommerville, I. (2001) *Software Engineering, 6th ed,* Addison Wesley, Harlow, England.

Standish, T.A. (1998) *Data Structures in Java. Addison Wesley*, Harlow, England.

SyncML (2002, October 22), *SyncML White Paper* [www document].
URL http://www.syncml.org/download/whitepaper.pdf

Timberline (2002, October 2002), [www document]
URL http://www.timberline.com

Tulachan, P.V. (2002) *Developing EJB 2.0 Components*, Prentice Hall, Upper Saddle River, USA

VoiceXML forum (2002, October 22) [www document]
URL http://www.voicexml.org

Whatis (2002, October 22), [www document]
URL http://www.whatis.com

Wiederheim-Paul, F. & Eriksson, L.T. (1991) *Att utreda forska och rapportera*, Almqvist & Wiksell, Malmö.

Ybuki, N., Shimada, Y., Tomita, K. (2002) *An On-Site Inspection Support system Using Radio Frequency Identification Tags and Personal Digital Assistants*, In proceedings of the International Council for Research and Innovation in Building and Construction CIB w78 conference.

Zarli, A. & Rezgui, Y. (2002, October 22) *A survey of Internet-oriented technologies for document-driven applications in construction open dynamic virtual environments*, [www document] URL http://cic.cstb.fr/ILC/publicat/zarli-rezgui-cibw78.pdf