Doctoral Thesis

Forthcoming database
A framework approach for data visualization applications

Author(s):
Mieusset-Kang, Kuk Hwan

Publication Date:
2005

Permanent Link:
https://doi.org/10.3929/ethz-a-005035824

Rights / License:
In Copyright - Non-Commercial Use Permitted
Forthcoming Database
A Framework Approach for Data Visualization Applications

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH
for the degree of
Doctor of Technical Sciences

presented by

Kuk Hwan Mieusset-Kang
Diplom Informatiker RWTH Aachen

accepted on the recommendation of

Prof. Dr. Gerhard Schmitt, examiner
Prof. Dr. Moira Norrie, co-examiner

2005
Abstract

Collaboration with information architects requires insights in aesthetics, information visualization, development processes, database use and user interaction. Based on this premise, the forthcoming database concept is conceived that frames the essential functionalities of the database according to the needs in building information visualization applications. It defines a new position for the database and serves to improve awareness and transparency throughout the application, including relationships among the users, application designers and the database.

Information visualization is a study on graphical presentation methodology that represents information characteristics including structure and relations in consideration of user interaction on it. Thus, its applications represent data so that it can be used for many tasks including data exploration, knowledge discovery and work space communication and collaboration. Users need to be able to integrate and negotiate their interests in the collaborative work environment, and application designers want to spend less effort when developing, adapting and maintaining the system. Furthermore, the database need to be promoted to support the visualization concept that the application designer utilizes as well.

The forthcoming database is a framework approach in response to these requirements based on our experiences in lecture and research. It considers awareness information, user interest, connectivity among system elements and users and communication between users and the application designer. In doing so, it provides functionalities complementing the application designer’s work, attracting user involvement, offering fundamentals for communication and promoting the database through automated processes and transparency. As a result, the database moved from a static position to an active one, with regard to visualization, due to its forthcoming behavior.

The implementation of the forthcoming database concept is accomplished as middleware consisting of several layers representing its functionalities. The system implementation was developed using the programming language JAVA and C++. For the database, ORACLE and MySQL are used. Programming languages and DBMS were chosen according to the application and the designers’ preferences.
The implemented concept was applied to three case studies, which served to prove and fine-tune the forthcoming database concept. Two of the case studies are web-based applications, an architectural course environment and a virtual office environment. The last case is an immersive 3D environment for data exploration. Each case study analysis is based on our own observations and designer and user feedback that was recorded and analyzed.
Résumé

La collaboration avec les architectes de l’information requiert de la compréhension pour l’esthétique, la visualisation de l’information, le processus de développement, l’utilisation des bases de données et l’interaction avec les utilisateurs. Sur la base de ces prémisses, on a conçu le concept de base de données prévenante, un concept qui décrit les fonctionnalités essentielles d’une base de données utilisable pour la construction d’applications de visualisation d’information. Ce concept définit une nouvelle position pour la base de données et sert à améliorer la conscience de la situation et la transparence à travers toute l’application ainsi que les relations entre les utilisateurs, le développeur de l’application et la base de données.

Les applications de visualisation de l’information représentent les données pour qu’elles puissent être utilisées dans beaucoup de domaines tels que l’exploration de données, la découverte de connaissances ainsi que la communication et la collaboration à l’intérieur d’un espace de travail. Les utilisateurs ont besoin d’être capable d’intégrer et de modifier leurs intérêts dans l’environnement de collaboration et le développeur de l’application veut consacrer moins d’effort durant le développement, l’adaptation et la maintenance du système. De plus, la base de données a aussi besoin d’être promue pour supporter le concept de visualisation du développeur de l’application.

La base de données prévenante est une approche répondant à ces exigences fondées sur nos expériences durant les cours et notre recherche. Elles considèrent les informations concernant la conscience, les centres d’intérêts des utilisateurs, la connectivité entre les éléments du système et les utilisateurs ainsi que la communication entre les utilisateurs et le développeur de l’application. En faisant comme ça, on obtient des fonctionnalités complémentaires au travail du développeur de l’application, qui attirent la participation de l’utilisateur, qui offrent des bases pour la communication et qui perfectionnent la base de données par des processus automatisés et transparents. Au final, la base de données n’est plus statique mais active au niveau de la visualisation grâce à son comportement prévenant.

L’implémentation du concept de la base de donnée prévenante est réalisée en tant que middleware composé de plusieurs niveaux représentant ses fonctionnalités. Le système a été implémenté en utilisant les langages de programmation Java et C++. 

Pour la base de données, on a utilisé ORACLE et MySQL. Le langage de programmation et le DBMS ont été choisis selon l’application et les préférences du développeur.

Le concept implémenté a été appliqué à trois études de cas particulier qui ont servi à prouver et à peaufiner le concept de la base de données prévenante. Les deux premiers cas sont des applications basées sur le réseau internet, un environnement de cours d’architecture et un bureau virtuel. Le dernier cas est un environnement 3D immergé pour l’exploration de données. L’analyse de chaque cas étudié est fondée sur nos propres observations et sur les compte-rendus du développeur et des utilisateurs qui ont été recueillis et analysés.
Acknowledgement

I would especially like to thank my advisor, Prof. Gerhard Schmitt for his continual support and encouragement. In introducing me to the CAAD world, he showed me another possibility to share my knowledge with others. His guidance played a key role in many stages of my work and helped me to grow as a researcher.

I thank Prof. Moira Norrie for being my co-referent and introducing me to her group. It was a pleasure to be able to discuss my work with her and exchange knowledge with her group. She always guided me in a very personal and warm way. Her expertise in computer science and constructive comments helped me to enhance the quality of this work very much, both in terms of content as well as writing style.

Being a research co-worker of the group CAAD and Architecture was fascinating and inspiring in many aspects. I enjoyed watching and assisting in the development of their beautiful ideas and working with them as colleagues. I especially thank Prof. Maia Engeli, Benjamin Stäger and Andrew Vande Moere for providing a meaningful case study environment and sharing their knowledge with me.

In CAAD, I came to know people who always supported me with faithful advice in many different situations and who have touched me personally: Mikako Harada, David Kurman, Nathanea Elte, Kai Strehlke, Mishka Bugajski, Marcia Pereira, Bige Tuncer and Rudi Stouffs. Especially, many thanks to Andrew Vande Moere and Arley Kim for their wonderful persistence in helping me keep a good morale for the work.

I would like to thank my family for their interest and belief in me.
## Contents

1 Introduction ................................. 1
  1.1 Information Visualization and Database ............... 2
    1.1.1 Information Architecture - Database as Space of Data .... 3
    1.1.2 Database - the Engaged Media .......................... 4
  1.2 Contribution ..................................... 5
    1.2.1 A Concept Combining Visualization and Database .......... 5
    1.2.2 Time-varying and Volumetric Data Processing in Real-time . 7
    1.2.3 User Preference and Event Modeling ...................... 7
    1.2.4 Communication and Distribution Channels .................. 8
    1.2.5 Case Study Applications and Evaluation .................. 8
  1.3 Thesis Organization .............................. 8

2 Background ...................................... 11
  2.1 Information Visualization .......................... 11
    2.1.1 Integration of Data and Display ....................... 12
    2.1.2 Flexible Architecture for Collaboration Data ............. 13
    2.1.3 Relation to the Thesis .............................. 14
  2.2 Computer Supported Cooperative Work and Awareness .......... 14
    2.2.1 CSCW: Computer Supported Cooperative Work ............... 14
    2.2.2 Awareness-oriented Groupware ............................ 16
    2.2.3 Relation to the Thesis .............................. 17
  2.3 Database Issues toward Information Visualization ............ 18
    2.3.1 Database-centric Visualization Environment ............... 18
    2.3.2 Active Database .................................. 20
    2.3.3 Group Activity Database ................................ 21
    2.3.4 Relation to the Thesis ................................ 22
  2.4 Summary: Tailored and Integrated Facilities for Information Visualization ................................ 22

3 Forthcoming database ............................. 25
  3.1 Definitions of the Terms ............................ 26
  3.2 The Necessity of a Framework ........................ 27
    3.2.1 Vision of a new Interaction Model ..................... 28
    3.2.2 Objectives of the Framework .......................... 30
### 3.3 Forthcoming Database: Framework for Active Interaction

- **3.3.1 Application and Awareness Object Handler**
- **3.3.2 Cache Handler**
- **3.3.3 Communication Channel Handler**
- **3.3.4 Comparison with other conceptually related works**

### 3.4 Properties of the Application Framework

- **3.4.1 Interaction Data State Example**
- **3.4.2 Collaboration Model**
- **3.4.3 Characteristics of the Application Framework**
- **3.4.4 Users of this Framework**

### 3.5 Recapitulation

### 4 AwareDB: Java-based Application Middleware System

#### 4.1 Design Choices

#### 4.2 Data Modeling Components Layer

- **4.2.1 Visible Database**
- **4.2.2 Interest Setting**
- **4.2.3 Comparison between Concept and User Interest**
- **4.2.4 Information Distribution**
- **4.2.5 Meta Tables and API**

#### 4.3 Cache Components Layer

- **4.3.1 Parallel Access in Shared Memory**
- **4.3.2 Preparing Data Sets**
- **4.3.3 Interpreted Data**

#### 4.4 Communication Components Layer

- **4.4.1 Information Transfer Protocol - Package**
- **4.4.2 Client’s Connection**
- **4.4.3 Communication to Appropriate Database**

#### 4.5 Integration by Java Applet Extension

#### 4.6 Summary

### 5 Applying Forthcoming Database

#### 5.1 Road Map of Case Studies

- **5.1.1 Three Case Studies**
- **5.1.2 General Paradigm of Existing Evaluations**
- **5.1.3 The Evaluation Directives**

#### 5.2 Awareness Support for web-based classroom environments

- **5.2.1 Goals of the Study and Data Collection**
- **5.2.2 Course Environment, Participants, Process**
- **5.2.3 Set-up for Information Display**
- **5.2.4 General Observation and User Evaluation**
- **5.2.5 Analysis and Design Decisions**
- **5.2.6 Summary**

#### 5.3 Virtual Office - Laboratory Design Experiment
Chapter 1

Introduction

It is time for a founded reconsideration about database use in information visualizing applications, and accordingly, new concepts are needed to modernize conventional thinking about database systems and their function. One of the main goals of this study is to address the inadequacies of the database form for information visualization and empower its active and collaborative role in the process of visualization application building. In contrast, many past research efforts, including active databases and different kinds of frameworks, still do not provide effective solutions that could remarkably change the position of the database according to users’ and application designers’ expectations of it.

The forthcoming database envisions joining the requirements of application designers with the capabilities of database systems by providing an integrated platform, a kind of framework for aesthetic information visualization. In fact, it aims to support the data exploring processes of the visualization applications and to allow the free movement of data in the whole system. The forthcoming database framework combines insights from the fields of database systems, groupware and information visualization as is described in Chapter 2. The important key words in the figure 1.1 from each area helped us come to a definition of the forthcoming database.

This thesis was completed through several projects strongly engaged in the topics of data, information, visualization, communication and collaboration in 2D and 3D environments. All the projects were connected to database systems, and consequently, this research focused on exploiting and optimizing the relations to database technologies: These include those between the database and its applications, database and application designer, and application designer and application user.

The fact that the forthcoming database concept is a result of collaboration with researchers in the area of CAAD (Computer Aided Architectural Design) led us to think about the aesthetic aspect of the data visualization and adopting a strong conceptual approach to essential themes. The resulting framework could be used not only for web-based environments, but also in general visualization applications.
It became a framework that could be applied for every application aspiring to provide data to users, taking advantage of improved database functionalities according to their visual and aesthetic concept.

In this chapter, the related motivations of this thesis are explained and the research contributions are listed.

Figure 1.1: Information visualization, groupware and database technologies are three backgrounds which the forthcoming database concept grew out of. The key ideas, including 3C (communication, collaboration and cooperation), transparency, aesthetic representation, and initiative behavior, helped us formulate the forthcoming database concept.

1.1 Information Visualization and Database

Database systems are increasingly used by non-experts, and consequently, they are no longer simply a technical issue to be solved by experts. It is becoming an essential part of many other areas, including art, work tools and the online shopping mall. To represent data meaningfully and provide a higher quality of user-data interaction, different scientific areas including information architecture, database technologies, groupware and human computer interaction are bringing their contributions together. Especially novel data visualization concepts are taking an important position as the foundation of user applications. The application designer of such visualizations as meaning-maker of data should be aware of many other issues, including user expectation, user action and database function, which require different kinds of knowledge.
1.1.1 Information Architecture - Database as Space of Data

In spite of the general understanding of architecture as creating physical space, the researchers of Computer Aided Architectural Design (CAAD) [Sch99] have extended this definition to virtual realms and successfully promoted the computer as an essential piece of equipment for their work. Moreover, they continuously produce new contributions in the research world and redefine the domain information architecture [Dav98, Eng01], in a way that reflects visionary inspiration.

Architecture has been a discipline for creating space [SWKvdM94] and its extents include virtual, cyber and information realms [Gib84, Eng99, Mer00]. Their intensive consideration of the new technologies such as the Internet, database technologies and architecture led to products such as “digital information landscapes” and “digital territory” [ES98]. In providing those visual environments, the information architect applied their architectural requirements including “usability”, “consistency” and “aesthetic qualities” in information visualizing applications as well [Eng98, EMB99]. To extend their great freedom in modeling, which might be sometimes difficult in the physical world, they invented intuitive space modeling tools [KEE97, SE01b]. These tools enabled them to create architectonic knowledge in cyberspace. In practice, these were often used as collaboration platforms among users in different locations and knowledge representation platforms where the users could submit their architecture-related stories [Str99, HGH+00, KSH+00]. There were also considerations in which the database itself was taken as a kind of cyberspace. For example, fake.space allowed users to link their individual contributions in the web-based community space, view other works, and navigate through the works in various ways. It was designated also as a “database-supported collaborative work” environment [Hir98, PT99], which was very committed to the database technologies. These applications always used a database system as a back-end system accessed by various programs, which are used mainly for implementation purposes in those applications.

On the other hand, there are applications where a database system is necessary, but often neglected and not mentioned. For example, scientific visualizations and computer graphics communities focus especially on visualizing abstract data, such as the human anatomy, and address the topic of user navigation. However, it seems that rendering and simulation work is being refined without overseeing the support of some important attributes involved. For example, time is considered to be an important attribute to be visualized in medical data [Chi01]: It is important to provide this information in an environment to explore a patient’s sickness history. As such, even though motion, interaction and time are becoming more and more important in the information visualization research area, database systems and real data use are still underestimated in the development of such computer graphics applications.

There are research domains where database involvement could be improved. Poor support for interactivity, aesthetic or performance for loading huge data amounts
might be one of the reasons why the use of database systems is not so widespread as we might expect. Also, it can be observed that database use is still not tailored for the casual programmer of information visualization applications, even though database systems nowadays can provide many advanced functionalities for well-experienced application developers. Through collaboration with information architects, the importance of the support of aesthetic and new kinds of information visualization is recognized, and it is believed that the requirements could be satisfied by evoking database functionalities. The information architects’ capability in “information processing aesthetics” (see [Moe04b] in page 3), could enhance many objectives of our life and, surrounded by data and database technologies, could support their work in ways that are different than what we are accustomed to.

\subsection{Database - the Engaged Media}

A variety of applications in science and commerce need the database systems as their system component. However, the database system is usually not visible to the final user except for the few people who use it for a specific goal such as programming. According to the spread of applications applying a database system, there are many efforts to bring the database content more meaningfully to the non-expert user and attract a wide range of users [CFS00]. Currently, the most popular way to access a conventional database system is offered through the Web. Here, several targets are taken, and data access is assured at a reasonable rate [Ern00], providing context dependent information [NP03] based on a well-defined information concept. This kind of database interface would be appropriate for a wide range of users including database experts and final users.

Casual users often see the database as a huge static container, and often they need to struggle to read out the small amount of data that interests them. Ironically, the application developers often find themselves in the same situation. Thus, to aid database exploration, most database systems offer an interface application that is designed for their specific database system. Unfortunately, these are still based on the endlessly opening of new windows representing objects or rigid-tables. To cope with this problem, there are studies showing other possibilities of data representation [Kur99, Moe01], providing practical tools on the whole [SKT+98] and a new classification model for abstract information [MB04]. Still, there are many technical hurdles to integrate such new database representations in an application independent of any specific database system.

Other studies make database activities, such as updating a row in a program, visible outside of the database itself. Using triggers or background programs inside the specific database system, it is a great enhancement of database capabilities to make this kind of information accessible [GJS92, HCD+98]. It gives us a vision of new aspects for databases that we are not yet trained for.
Scientific data visualization needs strong support by database systems. Powerful caching strategies and performance in providing huge amounts of data are needed, which is unfortunately difficult to achieve at the moment [RGC97] (see also in section 5.4.4). The further development of an active database geared to the visualization of information coming out from a database system dedicated to that purpose needs to be encouraged. In addition, intuition, interactivity, and aesthetics should be taken into consideration in combining these aims. Interactivity is a typical feature of information visualization applications. However, databases hardly accommodate user interactivity [LG95], thus seriously complicating application design.

In summary, the database interface has to have elements that support the visual quality of the data space for the database to properly function as an interactive medium. The integration of database facilities can tightly couple the database systems with information visualization to support the visual quality of the database as a whole.

### 1.2 Contribution

The research proposes the forthcoming database concept to couple database use with the visualization concept. A framework has been developed in which the features of a forthcoming database concept are designed and situated as a system for harmonizing operations. It groups all major components for a forthcoming database. This framework respects the necessity of a wide connectivity of systems, user engagement, a data metaphor concept, and awareness support in the system. Several stages and aspects of the prototype are presented. The main contributions of this thesis are situated in the area of information visualization and database technologies.

#### 1.2.1 A Concept Combining Visualization and Database

*Development of a concept based on the experiences in the area of database technologies and visualization applications. This concept outlines the specific needs of visualization applications with respect to data management and supply in a framework, named the forthcoming database. The main attempt of the forthcoming database is to consider how to bring the functioning of the database and visualization concept together.*

To address the issue of database systems and their role in information visualization, the expected characteristic of the database system or data supplying system is named as “forthcoming”. This thesis proved that the forthcoming feature is especially desirable for the visualization of dynamic characteristics of data sets, including reactivity with regard to users’ action and time-variability. In summary, the following requirements should be fulfilled by a forthcoming data supply system:
• Supply data with reasonable performance to the visualization application, so that user interaction with the visualized data sets is improved with speed.

• Allow the application designer to remodel the application objects without having direct access to the original data management system. Provide an interface with widened functionalities to access data sets related to the application in addition or parallel to the interface offered originally by the data management system.

• Allow the access to transform awareness data according to the application and guarantee access by the user and application designers.

• Employ visualization specific needs such as volumetric or time-varying data implementation to the mechanisms of data supply and efficiently provide the data accordingly.

• Let the application environment communicate with other environments in which the same database is used. Also, combine base communication methods and provide it as foundation for further development.

• Allow the addition of applications which enable multiple users to communicate with each other, offered as part of the basic structure for the application. In addition, provide functionalities so that the users can be informed by the database as well.

The forthcoming database concept combines the information visualization application and database role. It breaks the idea of the static image of the data supply system in that it provides collaborative and helpful in-depth functionalities and considers both application designers and final users in defining the needs of the database user.

Depending on the visualization concept, there are data sets that especially benefit from its use. Especially the visualization of motion, time, events are enhanced through the use of implemented forthcoming database concept. The visualization application designer and its users are already considered at the concept level and the forthcoming database proposes the better interaction and communication base.

In short, this thesis proposes a forthcoming database framework that is useful for data visualization concepts and their implementation because of its attentive support for the application designers’ requirements according to user awareness and data handling for specific data distinctions.
1.2.2 Time-varying and Volumetric Data Processing in Real-time

Development of data management processes for an application, in which the data values are time-varying and volumetric, and are easily influenced by user interaction. For such an application realization, the data sets should be retrieved in large quantities and in a reasonable time, so that application side implementation can be completed efficiently for frequent data evolution and the changing of data values through user interaction. This thesis proved that this implementation is possible using a cache components layer in the forthcoming database framework and at the same time, improved the forthcoming behavior of the data supplying side.

As the visualization metaphors or concepts differed very much from each other, different features of data get emphasized and thus, the manner of data management and supply as well. Implementation solutions for time-varying and volumetric data processing are shown in this thesis. The frequently changing data values stimulated active data supply of the framework to overcome performance difficulties. A forthcoming behavior was also stimulated: the preparation of data sets beforehand to predict which ones will be required with regard to the attribute set interactively in the real-time application.

Additionally, specific needs such as time-varying and volumetric data support proved the quality of this approach by providing the data sets in real-time to a working application in a 3D immersive environment.

1.2.3 User Preference and Event Modeling

Development of interfaces allowing access to the database schema to model users’ preferences with regard to the information they are interested in or to model event handling processes. These modeled preferences and event handling mechanisms serve as functionalities to improve the user awareness.

Several possible user actions on the database schema are enabled through simple interfaces that deal with personal preferences on certain data and the manner of information distribution. It enhanced the flexibility of data usage form even though this extent can be limited depending on the application designer’s preferences.

By means of modularity, modeling tools can be easily included in different applications and serve as their own application. In this thesis, a possible implementation using a simple button-based representation of a database scheme is shown.
1.2.4 Communication and Distribution Channels

*Development of communication foundation, which can be also used as channel for the information transmission such as distribution and exchange. In addition, it proves to be an advanced development platform for information exchange and extensions.*

For visualization applications, it is important to get data sets at the right place including frames and personalized area. This means that not only data retrieval processes are needed, also the channels which transmit and supply the data are important. This thesis proved that the supply of a communication base as an elementary system structure is rather a necessity and a forthcoming feature for visualization applications. An implementation of the communication foundation is shown that is useful for web-based applications and this also proved to be beneficial for extending visualization application domains. A connection from an application of a 3D immersive environment to an application of a web-based application could be made.

1.2.5 Case Study Applications and Evaluation

*Development of several applications, in which the forthcoming database framework is applied. These are web-based course and office environments and immersive virtual reality applications.*

To prove the versatility of the forthcoming database concept, the forthcoming database framework was applied in three case studies. Two of them, section 5.2 and section 5.3, are implemented and refined for practical use in a web application. The last one, section 5.4, was demo-oriented and for virtual reality application. Each of these applications focuses on a slightly different aspect of the forthcoming database according to the designers’ and program’s awareness requirements, and their analysis provided valuable findings for the development of the framework concept. The analysis categorizes the features of the forthcoming database framework as valuable for the collaboration among user, application designer and database. Further, its extendability is recognized as well.

1.3 Thesis Organization

This thesis starts with the main motivation to combine specific technological and cognitive findings into a behavior-related concept. This is followed by short descriptions of the three related fields which have inspired this work: Information visualization, database, and awareness in Chapter 2.

Chapter 3 investigates a conceptual framework, called the forthcoming database. First, the forthcoming database is defined, and then its necessity and objectives are
described using an interaction model. The architecture of the forthcoming database is proposed as a middleware framework containing layers with different functionalities. Finally, its properties are illustrated using concrete examples.

Chapter 4 is devoted to the implementation issues. Here, a prototype of the forthcoming database concept is described. This version is an implementation done using Java and partially using C++ to adapt the programming environment involved. Even though the prototype is built using Java and C++ programming languages, the framework principle should not be thought of as language specific.

In chapter 5, three case studies and their evaluations are introduced. It begins with a short description of the context in which the framework was applied. The first case study describes a course environment where the framework concept was applied first. In this study field, there were differences in the skills of the application designers and technical persons with respect to the application realization and the idea of visualization, which then inspired us to seek specific system behavior. Furthermore, we became more acquainted with the mechanisms of awareness in a web-based work environment.

The second case study, a virtual office tool was introduced. It involved the collaboration between an information architect and a technical person. The tool provided a virtual information space where people could get information about each other’s reachability.

The third case study describes the most recent example of where the framework was expanded for different languages and domains. It used the data visualization metaphor infoticle [Moe04a] in an immersive environment. This ultimately became a data exploration application of a large database. The size of information per object made the performance of the visualization very difficult. The database was frequently updated and its presentation done by the time varying behavior of the infoticles.

At the end of this chapter, tables summarizing the above case studies and an analysis of the results are shown.

Finally, chapter 6 concludes this thesis with a summary and gives a discussion about further conceptual possibilities for databases and information visualization research.
Chapter 2

Background

The concept of the forthcoming database started to take form during work on projects concerning information visualization. It involved building collaboration systems where quality was improved through appropriate data representation and user awareness support. Finally, the system served as a groupware with increased quality of interaction between user and data. In a way, the concept is the answer to many questions that arose through the course of this and earlier work. The questions were mostly related to information visualization and database approaches. The answers were sometimes found in other works in these areas. Even though the focuses of these works were different than this thesis, they offered material to be explored in further research on the forthcoming database. Thus, in this chapter, an introduction in the research topics that provide the background of the forthcoming database concept will be given. These are works in the areas information visualization, groupware and databases.

2.1 Information Visualization

There are many statements expressing the power and importance of effective visualization including, “A good user interface is as important as proper functionality” [Ros92] or ... about 70% of an applications code is devoted to user interface handling, and only 30% deals with the actual application” [Der90]. Not only through others’ but also our own experience, the impact of user interfaces on the comfort and efficiency of the work environment was noted. Accordingly, the information visualization community considered the issue of how to present information and invested in techniques such as multidimensional scatter graphs, information landscapes and spaces, node and link diagrams and trees. The information visualization environments also can be differentiated by data type and include documents (1D), geographic information services, newspaper layouts (2D), human body, building, molecules (3D), movie database, stock market statistics (multi-D), tree, network and workspace [RH97]. One dimensional data such as program listings are often text-based and thus, methods such as scrolling are a possibility for the user to...
reach the desired data element. The newspaper layout is a method to represent two-dimensional data types such as height and width in a two-dimensional visual environment. In case an application shows more than two attributes on a 2D display, it can be seen as both a multi-dimensional and 2D environment. The 3D data type includes real world objects and synthetic objects. These are mostly used as data types in volume visualization methods such as those found in medical applications. Multidimensional visualization represents data sets in which the number of attributes are more than three. Tree structures are mostly used to visualize hierarchies and their visualization is traditionally done in 2D representation. However, the data type of tree is similar to the multidimensional one with an intrinsic hierarchy.

A workspace is a specific multidimensional data type. It can be seen as a collage of data types of different dimensions. On one hand, this workspace visualization concentrates on putting more windows in an organized way and on the other, it modifies the environment supplying new information. The applications handled as case studies in this work are of the categories 2D, multi-D, network or workspace visualization. As the application designers’ point of view is strongly considered, topics such as how to integrate data and display, frameworks tailored for data extraction and transformation and architecture for specific collaboration data were taken into consideration.

2.1.1 Integration of Data and Display

Information visualization has entered the mainstream in order to allow the codification of art into technology. Under this statement, an effort to design new visualizations is called for.

According to Card [CM97], a framework for information visualization can be defined by means of three properties: data, automatic and controlled properties. The value of a data can be a nominal (equal or differ to other data values), ordinal in that the value obeys the relation of $<$ and/or quantitative. The automatic properties are graphical properties such as position. The controlled properties can be connections such as in a tree presentation and enclosure. The framework offers a possibility to analyze existing mappings of data and intent to the visualization. Further, it enables the abstraction of the visualization technology that can be applied for practice.

Another study of information visualization approaches seeks to generalize integration between data and display. A conceptual framework is proposed for all possible visualization types operators [CR98, Chi00]. The operators are defined as all possible user interactions with the visualization system that may change the data source (value operator) or only the visualization content (view operator). It clearly separates the value (raw data) and view (visualization) of operators and proposes a visualization operator model that takes the multiple value/view and data state description into account. In practice, it simplifies the end-users’ choice to get their
desired result and to predict the result of their interactions with the visualization system. The general model should help the designers in classifying and understanding the relationships between operator and the composition of interactions. The result of this framework enhances the understanding between users and designers.

The information visualization pipeline usually offers processes that transform raw data to a user view. It first forms analytical abstraction, which is often further transformed into a visual form before it gets mapped into the application concept. The effort to integrate the data and its view is also supported through a new architecture concept along the visualization pipeline, in that seamless transformations of the data from one end to the other are striven for.

To support the extraction of data and further transformation, Groth [GR98, GR02] proposed an architecture with the key element, data preparation component. Data preparation provides a mechanism for transforming data into a form that is necessary for the visualization. It enables the developer to define and implement the appropriate data transformations (user-defined map) for their application. Using a graphical interface, the user can access the original data, define a transformation map and implement it for their application. The final result is shown as Java 3D graphics. Thus, without an actual change of the original data, users can transform the data into their application data objects.

2.1.2 Flexible Architecture for Collaboration Data

Schönhage [SE97] proposed a flexible architecture that can enhance a system to a multi-user visualization one. To support users with different backgrounds and individual needs, he argued that the visualization has to offer multiple-views on the information. In addition, adaptable visualization to allow experimentation and a networked or web-based architecture to support visualization at the user’s desktop are further requirements. In short, it should consider the users’ personal preferences, experiences and easy access from the user’s desktop. The DiVA (= distributed visualization architecture) [SE00] regards the process of visualization as a transition of data. Thus, the architecture consists of three, primary, derived and presentation models and the two mapping processes among the models, conceptual and presentation mapping. The primary model generates all data that is needed for the visualization. This could be described as a selection of data sources. On the data of the primary model, the derived model processes their pre-programmed concept. In the primary model, there is a shared concept space based on hierarchical conceptual data. This derived model looks differently depending on the user even though all of the users have the same primary model. This enhances the adaptability of the system and could even dynamically accept user feedback. The presentation component is a library of visualization primitives that are used by each user to map their concept in the derived model. Also users can share the same derived model and have different presentations. For DiVA, there exists a CORBA-based prototype.
2.1.3 Relation to the Thesis

In the area of visualization, the necessity of a kind of fusion between the data and visualization was recognized. At the same time, effort was made to develop a general method for the visualization of different kinds of data.

For this thesis, the lectures of the CAAD and Architecture chair at the ETH Zurich, Switzerland, which contained several aspects of the information and visualization, opened the door to the visualization world. Additionally, the key terms data preparation and operators already presented in the works above led this thesis to define important system properties. The process of data preparation to supply data with reasonable speed to a 3D immersive scene was gained from the idea of the background work. As the web-based and networked system architecture was also recognized as an important factor in information visualization systems, this thesis concept was implemented as middleware. In some cases, such as the visualization in a 3D immersive environment, there are many restrictions with regard to the hardware and software needed, which hinders extensions or connections to other environments. In this thesis, however, we attempted to connect both environments: 3D immersive and Web environments, with help of flexible and portable architecture.

2.2 Computer Supported Cooperative Work and Awareness

The information visualization applications used in these projects were normally designed for multi-users, and they offered a kind of shared workspace to the user. Thus, it was essential to consider the aspect of groupware. The other way around, it is a very important fact that groupware can provide good information visualization to enhance collaboration among the users. However, the groupware systems often offer a window-based interface and do not consider many of the visualization aspects involved such as interaction or aesthetics. It was further recognized that the information visualization applications concentrate much on the enhancement of the users’ awareness in relation to the information offered to users. Keeping in mind, however, that the groupware rarely use a database system to handle information for users’ awareness, it is appropriate to give a short description about CSCW and awareness in relation to these applications.

2.2.1 CSCW: Computer Supported Cooperative Work

Computer Supported Cooperative Work (CSCW) began in the mid-70’s. At that time, the fields of Software Engineering (SE) and Office Automation (OA) focused on computer support for large groups and projects. In 1983, with the development of telecommunication and the requirement of project-level software, the Human Factors Society started developing single-user applications and interfaces. In this
phase, the products were poorly accepted because of the inflexibility of the tools and the isolated handling of the applications’ aspects. Until 1994, there was growing interest in small-group applications, whereby the single-user applications could be enhanced with groupware features. On the other hand, the integration of applications in one large system and a generic framework were aspirations attained with the help of new technologies such as Common Object Request Broker Architecture (CORBA) [Gru94a, Bur97]. With the explosive spread of the Internet, people started to think of the web as a collaboration platform [Dix97], and this brought about the idea of collaborative web-suites, which offer several applications together over a web-browser.

CSCW is generally understood as a research field that focuses on understanding teamwork and developing and evaluating collaboration systems. This led to a collaboration of multi-disciplinary research areas, including psychology, communication science and computer science. People often associate CSCW applications and groupware with a software product, but CSCW is more about designing principles and concepts. Coleman [CK96] defined groupware as any technology specifically used to make groups more productive and to support person-to-person collaboration. Therefore, groupware is a part of the networked applications environment. For example, the access to a corporate database through a network may not be made through groupware, but interactive or discussion databases may be part of a groupware application. The terms groupware or CSCW application are also commonly used for the tool for collaboration with others in supporting the communication, coordination and also cooperation. Nowadays, the term groupware is becoming obsolete, because the applications tend to have been already built in as base functionalities in a system instead of being built separately. This tendency lets people say that all software will eventually become groupware.

The category groupware includes E-mail, calendaring and scheduling, electronic meeting systems, real-time data conferences, non real-time conference systems, collaborative authorship applications, workflow products, distance learning, telemedicine, Computer Assisted Software Engineering (CASE), Computer-Assisted Design (CAD), MUDs, etc. To classify these groupware, there are several methods including classification by degree of support according to collaboration, coordination and cooperation [Wen96]. However, the mostly used classification axes are time and place. Depending on time and space, the groupware can be synchronous or asynchronous and co-located or remote.

In this thesis, the term groupware is used for applications that can provide necessary information for common affiliations and stimulates the communication and collaboration providing an appropriate interface. For the suitable information representation on the interface, the application also has to consider user interactions with the information and aesthetics to improve data recognition. In similarly conceptualized groupware, the importance of insensitivity to group qualities was pointed out.
Furthermore, the need of the designer’s understanding about user workspace was stressed by the researchers to recognize the magnitude of the problem and avoid the common assumption of a rational work environment [Gru94b, Gru94a]. The application designer’s effort to make the users aware of their information could be also observed in each application developed during this work.

### 2.2.2 Awareness-oriented Groupware

Perhaps because awareness is a natural and self-evident phenomenon, people rarely think about it. It is generally defined as knowledge or consciousness and has been studied by human factors researchers under the rubric of Situation Awareness (SA). There are many definitions for Situation Awareness (See in [Sar91, Fra88, Fra89]), and it is generally accepted as the following:

- the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future. (Endsley 1995)

- an understanding of the activities of others to provide a context for your own activities. (Dourish and Bellotti, 1992)

The topic of awareness has received a great deal of attention in recent work of CSCW, and there are several trends in CSCW research involving awareness support. First, multifaceted prototypes are made to study the requirements through development and evaluation. Secondly, a generic and flexible model is developed for awareness support and its notions. However, due to the awareness definition, which is not unique, it produced a range of issues considering particular contexts. These include workspace-, activity-, mutual-, group-, contextual- and peripheral awareness, which overlap in most cases.

In the following segment, several investigations on the topic of awareness in groupware are noted. NYNEX Portholes [LGS97] use video-based background awareness. Automatically updated video images of users in a community show recent movement and their presence. AROMA [PS97] uses abstract representations as a presence indicator for awareness design, because they believe it helps the user somehow to maintain their privacy, and it is easier to map to other media. The AROMA group defines their focus as peripheral awareness. This means the ability to maintain and constantly update a sense of people’s social and physical context. Tollmar et al. [TOS96] defines social awareness. In other words, the concept revolves around the social situation of the members, i.e. awareness about what they are doing, if they are talking to someone, if they can be disturbed etc. AROMA provides a @work site where people see a research laboratory group and their situation to investigate the problem of how awareness should be conceptualized and how it can be provided for CSCW systems. As result of the study, they recommend including emotional
state and group members’ knowledge in social awareness. Further, the use of dual-purpose design was proposed. They defined it as a design for solving a current problem in work practice and offering solutions to enable new forums and new media for computer based communication. Gutwin [Gut97, GG98] defined the concept of workspace awareness in order to improve the usability of an environment for real-time collaboration. His work concerns the context of the workspace and technical issues answering artifact manipulation, view representation and workspace navigation and as a result, to enhance groupware usability.

Some of the works are based on a spatial model supposing that well-defined objects and relations among objects are the representation of spaces. The Aether model [SBB97] defines awareness as a joint-product of focus (how I direct my attention to you) and nimbus (how you project your presence or activity to me). This model offered an awareness engine that provides applications with necessary information, and it treats all the objects and their relations without making assumptions about the kinds of things the object represents. Rodden [Rod96] presented a model for shared cooperative applications, such as workflow and shared databases, representing the space as graphs of interconnected objects. [Rod96].

The concept of the activity awareness model is based on events that are a representation of awareness information. In general, an event is a formulated result of a user’s activity with respect to the objects, such as documents within a shared workspace. BSCW [BAB+97] refines a similar model, in which the events are triggered whenever a user performs an action in a workspace, such as uploading a new document or downloading an existing one. The model is responsible for the production and distribution of the event to the applications. Along this model research, the concept of activity awareness is defined [NHHG98] and further extended to the concept of interest profile [Fuc98] of each of the users, in which the user’s interest on events of a specific context can be specified to guarantee that the user gets informed about future events on a context.

Among these model concepts and systems are different trade-offs in terms of their flexibility, the modeling of personal interest or the structured procedure for users. However, the integration of models proposes a viable alternative by combining the advantages of several models [Bür99] or by adding an active data sharing capability among different applications [PRT99].

2.2.3 Relation to the Thesis

In the CSCW field, the awareness topic gained much interest of the researchers. Their studies are concentrated on the wide spectrum of awareness definitions with regard to groupware, the needs of specific awareness and their applicability to CSCW tools. Although, many case studies and examples on awareness can be found in the
rubric of CSCW, little has been implemented using real database systems. Consequently, finding a description of awareness that could be used in programming with help of databases is relatively unexplored. Although the study on awareness mostly occurred in a form of groupware, and groupware considers also the problem of information visualization, little attention was given to the general visualization metaphor. In this thesis, all the case studies used belong to the category of groupware and concern the definition of awareness supporting functionalities that are appropriate to user expectations of each application. Awareness was also a theme regarding information visualization concepts.

2.3 Database Issues toward Information Visualization

The purpose of the graphical user interface of most database systems is to assist schema design, query and browsing or manipulations [KT92]. In general, these graphical interfaces placed on top of a database system can be simple data browsers and/or serve as a toolkit for an interface builder. In the latter case, the application designer is asked to have knowledge about the database systems and is obliged to work with a specific database system. The work in the database research domain, specifically related to the visualization, can be categorized into two major directions. One is to bring information from the outside to the inside of the database to make the database more transparent and better accessible from the user’s viewpoint. The other direction is to bring the information of what happens inside of the database to the users outside.

2.3.1 Database-centric Visualization Environment

User interfaces of database systems demonstrate the effort to provide easy access for the users. However, learning query languages or formulating complicated data queries are still difficult for a non-expert. Thus, a system environment is offered by Aiken [ACSW96]. In the system of Aiken, one can use most programming operations manipulating graphical representation of data or programs represented as dataflow graphs. It provides a set of operations including Add, Set, and Swap Attribute(s) to facilitate building database applications. It also defines its own displayable data types, which means that these types have a calculated form of visualization. It uses the input visualizations and the sum of the input is the resulting value (tuple) visualization. Furthermore, it allows program operations for graphical programming, operations for rendering and offers several ways of viewing the data and its results, such as drill down or zoom.

To support more systematic and efficient development of the application interfaces to the database, model-based techniques have emerged in recent years [GMF+98]. They help the database interface developer’s work in providing diverse models, which
include the domain, task, user, dialogue and presentation models. Even though the idea of the model-based concept is aimed more for database interfaces, the framework seems to be useful for the developers of information visualization application as well. Especially presentation models describe how the information gets represented to user and this can be specified by application designer with help of provided and extendable templates containing graphical primitives. As the presentation model has a strong relation to the application data structure, such a system reconstructs the user interface, in case the application data structures get changed.

The above systems can be described either as browser tools of a database system or as graphical database management systems with limited functionalities for application development. They differ strongly to ordinary database systems in that they allow direct graphical manipulation to simplify the programming and considers the non-expert’s aspect in their system by using the notion of big- and small programmer. The advantages of these systems include the capability to provide an open architecture allowing developers to reuse existing data visualization, schema representation widgets and component libraries.

However, these systems should be concerned to allow large amount of choices on primitives for all kinds of presentations or guarantee great freedom for developer in specifying application specific primitives. User-data interaction can also produce unexpected effect similar to self-organizing data sets in a spatial representation. This kind of specific data types could be also explored as primitives.

With respect to web applications, Web Modeling Language (WebML) [CFB00] offers a model based approach to enable high-level descriptions of web sites. In WebML, a web application can be produced with help of models that contain the descriptions about data content, pages composing the web site, links between pages, layout and graphic of the site and customization features of a specific person or group. Among them, the “composition model” provides six units to describe different hypertext and the navigation model expresses how pages are linked with each other.

The WebML is independent from client-side languages and server side platforms. The idea of having such an independent system fits in well with the concept of this thesis. Especially the “operation unit” [FP03] could serve as a base to describe processes with regard to users’ interest. Also, the high-level description which can immediately produce the web sites is a great advantage for a web application builder. However, the work with units could become rigorous for an application developer, because the effort involved in defining the units can easily become an overloading factor in the application development process. In the extended version of WebML [MCB+03], several additional units are defined as primitives for web services. It also shows that care must be taken with the possibly fast growing number of units. In addition, it might be also useful to offer a kind of base for applications supporting communication among users in addition to the units.
2.3.2 Active Database

The term active databases was first used by Morgenstern [Mor83] to denote databases incrementally updating views. Later, the application areas of active databases have come to include, for example, triggering facilities, constraint checking, etc. Today, an active database is defined as a database that is able to react to events and conditions in the database, or even the surrounding applications and environment [Eri97]. The active database system is also seen as a trend to bridge the gap between knowledge base and database systems [VB98] or to “trigger” procedures.

Active databases emphasize the notion that a body of information is dynamic and should respond intelligently and in non-trivial ways to the user. The active database provides a paradigm for uniformly addressing the information handling activities that are central to one’s computer work environment. The activeness of a database is a behavioral metaphor that involves several dimensions and emphasizes the dynamics of the user’s interactions and the desirability for system intelligence in dealing with the consequences and implications of these interactions. This concept emphasizes the notion that what the user sees instantaneously reflects what exists, including the changes and consequences of these changes.

In contrast to passive DBMS, which execute queries or transactions only explicitly requested, the active database serves applications requiring active and time constrained processing [Cha89]. Emphasizing the reactive behavior of DBMS, Dittrich [DG96] defined the active database as follows:

A database system is called active if it recognizes some definable situation in the database and releases some specific reaction additionally to the usual DBMS capability.

As such, an active database system is also understood as a conventional, passive database system extended with the capability of reactive behavior. The desired behavior of the active database system is expressed using event-condition-action or production rules.

- Production (in expert system technology) or “data-driven” rule

  IF condition THEN action

This condition describes data states that should be reached by the databases. This is a declarative description of a firing situation (a query) without an exact definition of how or when this situation is detected. When the condition is satisfied the production rule is fired (or triggered), and its set of actions is executed against the database.
- Event-Condition-Action (ECA) or “event-driven” rule

ON event IF condition THEN action

This rule is triggered when the event of the rule has been detected, either in the database, caused by a data manipulation operator or externally by another system. Only the condition of the rule is checked and if satisfied, the rule action is executed. ECA rules are understood as the most natural choice for generic rule support since events conform to the message-passing paradigm of object-oriented computation and every recognizable message/method can be a potential event.

Some researchers emphasize the integration of multiple rule types in the same system as being important to provide a single, flexible, multipurpose knowledge base management system [VB98]. For the practical implementation, a system component, or active component, to deal with these issues is needed.

Although it is very convenient to make the database system active, Dittrich mentions it may be a disadvantage to put all the mechanisms inside of the database system, because the DBMS then becomes bigger and bigger.

The trigger is certainly the core of the active database. However, the converging interest of active database and this thesis is more the idea of making automated procedures raising actions. Active or interactive behavior also became one of the main considerations in the field of information visualization. Thus, the idea from the side of database systems for being active inspired this thesis to attempt the integration of awareness and databases.

2.3.3 Group Activity Database

Group Activity Database (GADB) is a database concept proposed by Tarumi and al. [TMK99] and its functionality is proposed mainly for the analysis of groupware. This database is originally thought to be suitable for agent-based groupware. GADB assumes that an interface agent per user has to be built to make a user model. Then, the communications among agents and users are collected into GADB as a “capture log”, which indicates the history of communications and work processes. This log data includes, for example processes, workers, deadlines or all task requests that a given worker has at a given time or at a given time in the past.

To support the analysis, GADB provides a main simulation function of business processes. Using the log data in the simulation, a user model can be newly constructed, and through the new model it is expected to evaluate the impact of the groupware introduction and evolution to the resource (people). Even though the problem of reliability is present, this will help groupware designers and users to
improve the quality of abstraction of user behavior, to remodel processes, and to support team awareness by visualizing the log data about status of members and communications among them.

The strength of this database is its ability to recognize and visualize the bottleneck of groupware through the simulation. Therefore, the large amounts of possible log data are available, but the distribution of this data is still static. That means if an application wants to know about new log data of a user interface agent, the application has to send a query to the interface agent of the user instead of just being informed automatically.

2.3.4 Relation to the Thesis

The effort to stimulate the database as an active system fits in very well with the idea of this thesis. As the case studies in this thesis also showed emergent needs for improvements in several aspect of the data supply, we first moved the position of the database more to the front in the entire applications to find better solutions especially for visualization applications. This thesis proposed the term forthcoming to summarize the key characteristics of a database that were also considered partially by active database or under the term events. The data-driven rule is implemented into the concept for this thesis.

2.4 Summary: Tailored and Integrated Facilities for Information Visualization

Information visualization, groupware and database technology fields have shown similar and not mutually exclusive concepts such as suitable data representation and enhanced user-data interaction. In developing applications with the goal of providing efficient data visualization for users without overloading their perception and application designer’s work, the visionary concepts of three fields needed to be integrated into one system.

Tools such as the DBMS browser serve a class of developers well. But it can frustrate developers by confronting them with the learning process needed to get acquainted with a specific database system and the additional tools offered with it. Such tools are often designed as a toolkit for expert programmers and provides a traditional programming interface. However, how to specify some aspects of visualization except via ordinary statement and expression oriented programming has been considered less. On the other hand, there are many advanced functionalities in using database systems that seem to be not yet widely practised. The case studies of this thesis were not exceptions, due to the restrictions such as the specific development environment, compatibility, and the mixed focus of the research. In development of information visualization applications in which this thesis was involved, databases
are often used only for data selection, and the operations for the visualization are processed over retrieved data subsets. These are some of the reasons why a kind of independent middleware system was striven for. It is more flexible for inclusion in an application system and extendable rather than providing a fixed huge database system each time.

Information visualization is understood as a foundation of any application provided to any user. It is desirable to have a system that can sufficiently relieve the application designer’s work and gives the possibility to bring the data in the interface without struggling with technical knowledge. Thus, the data support should consider the application designer’s work, the interface concept and the user’s expectation at the same time, as well as their connection with each other.

In short, the background research areas provided helpful concepts and research directions for the thesis work process and framework - the forthcoming database. It gave motivations to design a tailored system specifically for information visualization that still maintains all the functionality integrated in a collection of modularized structures.
Chapter 3

Forthcoming database

In information visualization, one encounters many methods developed for data representation and user interaction. These systems have the tendency to be offered as a kind of group collaboration environment or as a tool for large data set exploration [GRKM92]. In collaborating with an information visualization community by providing technical support with data handling, it was notable that the actual requirements were sometimes quite different from the general understanding and expectations of such environments. In general, when applications are relatively easily accessible, such as web-based applications, there are many transactions with its databases. Otherwise in slightly complicated applications, such as Virtual Reality or immersive environments showing scientific visualizations, the majority of database systems are used only for data selection. In both of these cases, the processes for the visualization of data items are operating mostly over retrieved data subsets.

On the other hand, database systems tend to offer a kind of interface to access the data with data browsing functionalities or tools such as interface builders. However, these do not seem to have gained much interest among application developers. We observed that these additional tools had the potential of overloading the work of the application designer in terms of the knowledge that they had to learn. As a result, the application designers tended to prefer using only the core functionality of database management systems and load the work as much as possible on the application side. Ultimately, the application designer could not profit from the functionalities designed for their comfort.

This thesis seeks a concept that can propose an efficient support system to facilitate the work of databases and visualizations and tighten up the relationship between them. It is believed that the application designers’ work has to be considered more with respect to database use and a specific visual concept such as a newly invented visualization metaphor, which can produce unusual data behavior. Especially among such researchers working in areas where aesthetic and visually pleasing qualities play an important role, we as technical persons feel like and tend to be treated as strangers. Thus, in this thesis, it is emphasized that the collabo-
ration among visualization application designers and database management experts should be seriously considered in the framework design. Finally, the concept should provide comfortable access to the data, with respect to how the people involved interact with the system, including their vision, the visualization concept, and the application designer’s intentions.

In this thesis, the concept of the forthcoming database is built up as an application framework and further developed through experience with three projects: A web-based course environment, a virtual office, and a data exploration environment. In this chapter, the framework is defined and the architecture and properties of this framework are discussed. The framework developed here enabled us to define the relations among three subjects, the user, the application designer, and the data, to more efficiently work with data tailored for these applications and help the application designers in their implementation process.

3.1 Definitions of the Terms

To clarify the descriptions in this work, some terms are defined below.

A **visualization application** is a program dealing with information visualization and contains the end user graphical interface. It has its main focus in visualizing the data with regard to user interaction and information aesthetics and uses databases in its system. It describes a system representing data sources for specific functionalities including data exploration environments and collaboration environments. In the context of this work, an **application** is used synonymously with **visualization application**.

An **application designer** is the person who creates the application from the concept of how to represent the information, how users might perceive it, and how to realize it. Thus, the work of the application developer may include aspects from designing to writing part of or the entire code of the application and the work area that overlaps between designers and programmers.

**Application data** is data that has already been transformed for visualization use. It might have the same structure as the raw data. However, in most cases it differs from the raw data in fundamental ways.

**Awareness data** is data that is intentionally and with hindsight defined by the application designer to be used to enhance the user’s awareness while using their visualization. It is a subset of the application data and can be compounded from raw data sources. So, the information gained through the cognition of awareness data is called **awareness information**.
To avoid confusion, the term *user* is used only to refer to the end user, i.e. the user who interacts with the graphical interface visualizing information.

### 3.2 The Necessity of a Framework

To point out the essential implementation steps expected during an application development process, a scenario of an application designer’s work will be introduced. Going through this scenario, the important requirements will now be illustrated and commented on.

Take a look at a visualization application used by people working together from different places. It shows their shared documents and co-workers’ availability. Now let us think about the work of the application designer. She decided the look of the user interface and how to present the information inside of it. For example, she decided to map the “person” entity to a circle and a “document” to a square.

- To map the information with its content and add functionalities for user interaction with the visualization elements, she needed to retrieve data from the database. Using simple queries that she learned, a portion of data was retrieved and successfully mapped. Furthermore, she could retrieve data so that transient objects could be visualized on the screen, for example, a “change” object containing descriptions about the “person” and the “document”. However, it was difficult to then store the user interaction on this “change” object, because its structure was not one defined in the database schema. Programming work on the application side using several database queries was unavoidable, and the basic incompatibility between database object and application object turned out to be a problem to address.

- The application designer had a novel concept for visualization. After having each data object visualized as circle or square, she wanted to visualize each attribute of an object transformed to a stand-alone object. For example, instead of having the chapter number of a document as a simple number inside of a document object, new objects were needed as many as the number of chapters. For a document with 10 chapters, 10 new small squares might be represented on the screen. An attribute is a property of an object, and usually it is a single value item and not divisible. This might be specific to the application concept and thereby difficult to support using a conventional data schema. The common notation describing data seemed to be inflexible with regard to the visualization metaphor context.

- The application designer wanted to supply the visualization with a large amount of data very frequently, requiring the application side to wait. Consequently, the visualization would freeze a few seconds until all the necessary data arrived and was transformed to the objects on the application side. The
low performance of this data supplying mechanism disabled the comfort of the visualization, and user interaction was compromised.

- Since the application was provided for more than one user, the application designer wanted to offer the users a possibility to send each other information in that specific application using the data of their centralized database. However, this would be normally programmed by the application designer on the application side using her programming knowledge and specific language. To overcome the problem of disconnectedness, the application designer had to program first the communication mechanism.

- Through the user interaction with the visualization, many transactions were initiated to update the database state. For example, a new document was created and this could be informed to the other users before being retrieved and finally recognized. This informing mechanism could be programmed as a trigger or method of an object class, but this kind of work was too expensive for the application designer. She preferred to have another way to model this from the outside instead of having to access and change the core database. Consequently, it was expected to model the mechanism as an application side object using the database schema and store it as a permanent object. To reinforce the system with active behavior, she had to produce large code programs.

There might be many other scenarios describing the unlimited imagination of visualization designers and the inconveniences in using a database and other applications for those visualization works. The more the visualization concept broke with convention, the more difficult it was to bind it tightly to the underlying systems. Also, using a data browser or interface builder to merge with the visualization application turned out to be an expensive procedure for application designers. Thus, the application designers’ work with data should be supported more from the database management side, offering a founded concept instead of leaving them with patchwork programming tasks. Without new concepts integrating the necessity of the visualization with database functionality more tightly together, the work of the application designer might end up as long and repetitive, with interactions going back and forth and proposing an endless learning process regarding technical issues.

3.2.1 Vision of a new Interaction Model

Even though the above scenarios do not cover all of the cases personally experienced while developing applications, some general observations were made that are similar across the board. Building a visualization application requires a database management system, a visualization concept, and facilities providing programming possibilities for the visualization as shown in figure 3.1. The application designer should handle all three topics in addition to user interaction, which is one of the main focuses of visualization applications. Database software and its applications offer good functionalities for easy access. But the database is still not attracting the
user or application designer for this kind of interaction processes, and also it does not provide tailored response and support specifically for visualization concepts. Finally, the users and application designers do not feel that they are helped by database functionalities in doing their work. Thus, a more productive and positive interaction, one that makes the database more forthcoming to the other components as shown in figure 3.2, could be proposed. This changes the database from being accessed and passive to a system using and interacting with the other components. This is what this thesis suggests under the term “forthcoming”. It is common that

![Diagram of interactions](image)

Figure 3.1: Interactions happening during application development among visualization, database system, application designer and users. The application designer is the most active subject in the interaction.

![Diagram of forthcoming interaction](image)

Figure 3.2: The database services act as active elements in interaction with functionalities simplifying and facilitating the application designers’ and the users’ work.
database management systems offer their own data browsing possibilities and other helpful functionalities for data visualization. In spite of these complete offerings, in size and in functionality, they do not seem to be effectively used nor enduring due to the amount of knowledge and steep learning curve required. It is even less appealing to the application designer due to the passive behavior of this kind of software. Consequently, we would like to imagine a database system working in a forthcoming way. Its forthcoming behaviors encourage the use of integrated cohesive solutions for application designers, offering better functionalities for their work, informing the user about new states and changes in the environment from the many automated programs possible, helping the visualization provide high speed data streams, etc. We call this simple notion of a database system, which seems to be characteristic for the contemporary database system used to support interaction, figure 3.2, the forthcoming database.

3.2.2 Objectives of the Framework

The framework objectives derived from the aforementioned needs are to enable and enhance the following issues of the forthcoming database:

- **transformation and adaptation of the application data**
  It is desirable that the data sources in the database have exactly the same structure that the application designer wants to show on the graphical interface. It would be even better, if the result of a user’s interaction with the data could also be formulated using the same data structure as in the database. Application designers strive for almost perfect database schemas so that they can maintain them from the beginning to the end of application development. As already known, this is actually not the case, and there seems to be a large gap between the logic of the visualization designers and that of the database experts. Thus, the data retrieved from the database should be transformed to the expected form. This can be very exhausting work, since the data could be a composite of different data object classes. It is also very complicated when the result of an interaction is to be saved back in the database in a different object class. Thus, a process of simplifying the transformation and adaptation of the data is needed.

- **defining and structuring awareness data**
  In every collaboration tool, data sources are predefined, produced and continually changed as they are used. In order to keep people up-to-date with what is happening in their work environment, some of this base data can be carried over entirely as data for awareness information. Some other data sources possess only parts that are important and used for creating new data entities for awareness information. Therefore, the data sources should be definable as data for awareness information, during and after the development of the system. On the one hand, these should be handled separately from the data
of the application. On the other hand, they should be able to be considered as data entities such as application data. This will reduce potential confusion in the application developers' work and allow more flexibility in development and use.

- **easy extension and integration of applications**
  Due to the diversity of possible contexts and applications, the functionalities of the framework should be easily extendable, adaptable to a specific application environment and serve as a bridge between them. Nowadays, there is a preference to integrate the visualization architecture with the World Wide Web. The Web is an excellent medium to provide the context of the visualization, consisting of information such as text, pictures, sounds and video. By combining the Web and networked visualization, a more generalized vehicle to support people in making the system more easily extendable could be even obtained.

- **connecting users and system in an overall communication network**
  Since an application providing data visualization is designed for data exploration and as a work or play-related application, it could easily serve as a communication or collaboration platform. In doing so, the application must be able to accommodate many forms of communication - between, to, and from systems and people, and from one person to another or others. Messages must be able to be addressed to certain components in the environment, including people or databases. For this networked communication, the foundation of the overall connection among them is incredibly important, because it establishes the basis for the information distribution process.

- **accessing data sources from different perspectives**
  Usually, the application data model is fixed at the beginning of the development process by the application developer. He or she decides which data will be shown or not shown to the users as well as how to represent the information. In case the application developer and users do not have frequent contact with each other, it is difficult to make a system which is always up-to-date and thus effectively usable. To overcome this problem, it is important that system developers are kept informed of users’ interests, their conflicts with the information that is being offered and personal preferences. Often each user in a system of organizations has different access rights to information. Similarly, the focus of their interests may differ as well. Therefore, users should be allowed to describe their preferences with respect to certain data.

In brief, there is a need to construct a general framework - a conceptual model that clearly defines and organizes all necessities for building and maintaining a visualization application. The motivation can be distilled into the interaction model as shown in figure 3.2. As the needs indicate, the database system should be different than it has been until now. It is necessary to enable the database to provide
functionalities more actively using automated processes and to formulate the data according to the visualization requirements.

3.3 Forthcoming Database: Framework for Active Interaction

With respect to the above interaction model, a new concept, the forthcoming database, is defined.

The forthcoming database is a database system for visualization applications with functionalities for application-dependant objects modeling, awareness data support, interconnectivity network, and high-speed data supply. This kind of database shows forthcoming behavior.

Forthcoming behavior includes the preparation of data before the request is made, data transfer according to the user’s preference, and event data distribution without demand, procedures which are very much related to user awareness. It is similar to when the user would have a personal consultant in the normal language sense: The database is ready and waiting to help and give information.

In short, the forthcoming database is a concept that promotes database systems as the active interaction component for visualization applications. It provides an integrated solution including data modeling functionality from the visualization aspect, a networked communication basis, and the performance enhancement of data supply.

The forthcoming database offers assistance to deal with the problem of poor database performance with regard to the visualization applications and to optimize interactions during the application building process. The key design goal of the forthcoming database is to support the development of different kinds of data visualizations capable of visualizing data sets according to user interaction and aesthetically providing active processes and the basis for data handling. As described above, it encourages a new kind of active and forthcoming behavior in database systems by supporting interactive components of data visualization applications.

The practical experience in this thesis in making visualization applications that are mostly served as Internet-based collaboration or virtual reality data exploration environments provided a thorough understanding of the special needs involved, including reducing the repetitive work of application developers and difficulties in using data sources to support user awareness. Based on these observations, the approach of this thesis connects data sources, application developers, and users with each other in a way that is able to use data sources better.
To provide this concept as a system, a middleware system is proposed as a framework. The necessities and tasks are grouped together as components of the framework. The framework is composed by three main layers: the data modeling components, the cache components, and the communication channel components in a single cohesive middleware. To maintain flexibility among different applications and database management systems, the framework functions as middleware between the application visualization and the databases as an integrated system platform. In this middleware, the forthcoming behaviors of the database should be incorporated as much as possible to make the interaction model operative. The framework could be extended by adding a further functional inner layer to be used as a base system for developing applications, such as web-based visualization. The figure 3.3 shows the architecture of the framework that will be discussed in this chapter.

Figure 3.3: Architecture of forthcoming database framework containing three component layers: data modeling components, cache components and communication components
3.3.1 Application and Awareness Object Handler

Application Data Handler

Having an application data handler, the application designer could comfortably model objects in a way that might be useful for the visualization concept.

- **Application Object Modeler**
  Some of the application data and their structure are defined clearly separated from the database model using a kind of database schema browser. These concern the objects that are not of the same structure as the original database schema. It should be defined on top of the database schema, but this does not necessarily need to be done in the database software system environment or inside of the database. In this way, the objects will be mapped to the visualization metaphor faster and the modeling application might be more portable for different databases or applications. These kinds of modeler seem to be the simplest and, in general, the simpler the better to be included into other systems.

- **Application Object Transformer**
  There are differences of structure between application objects defined by the application designer and low level data objects. Thus, the transformation of the retrieved data to the object for the visualization mapping needs to be processed. In the application object transformer module, the process can be pre-programmed to take place whenever an application object is defined by the application designer. Another possibility might be a module responsible for comparing the structures on both sides and creating the transformation code automatically. This could make the changes of the database system easier.

Awareness Data Handler

Many data visualization systems aim to offer a collaboration environment. The system shown here is no exception. However, the case studies in this thesis belong to those, which finally provide both a work and a collaboration environment for the user. In such systems, the application designer’s intention to make the user aware of their data, environment and each other is remarkable (see also section 5.2 and 5.3). The lack of awareness support in a collaboration environment can negatively influence the connection and communication among the actors as well as the use of data sources. It promotes static behavior in these environments, because the information which may be helpful to people working together cannot be passed on to the users. This prevents actors from perceiving awareness information, a determining factor in the usefulness of the working environment. This weakness can significantly hinder the collaboration as a whole. Past attempts to optimize the collaboration environment in relation to user awareness have concentrated on improving graphical user interfaces. However, these approaches do not address the application developers’ efforts well and may not fulfill all of the requirements for implementation,
because technical sources including data structures and networked communications are not taken into account. Therefore, the problem of awareness information handling should be worked out through system design and implementation processes, not only through graphical user interfaces. In brief, this module is responsible for processes related to the creation and manipulation of awareness data, and these are divided into the following handlers:

- **Awareness Data**
  Some information is important to be perceived by the users for their interaction in an application environment. It might be the change of a specific kind of document or the presence of certain people. Important is that the application designer decided to show this information through the use of a remarkable representation, which improves user awareness and stimulates their engagement. However, all of the work to reassemble this information after retrieving and re-retrieving out of the resources might not be completed in the right way. Better is that the application designer can define this awareness information using the original database schema from outside of the database systems similar to the application object. For example, a new awareness data instance with name *presence* can be created with the attribute *name*, *telephone number* of the table *user* and the attributes *room number* of the table *office*. Also, it might be totally independent of the database schema. In contrast to the application object, this should also contain a description of how it could be distributed in order to reach the user. One category of this awareness data might be *event*. Event is an expression of captured users' interactions occasionally arriving to the database and even changing the database state. Thus, awareness data, including event, should contain the information about to whom and how it should be provided. This is the work guideline of the application event distributor.

**Example:**

```plaintext
// AN OBJECT OF AWARENESS_INFORMATION
Awareness_Information is:
  Identity
  Description of this data
  Users who is interested in awareness_information
  Action description
    The possible manner how it should be delivered to users
end Awareness_Information
```

- **Personal Interest Handler**
  This handler allows users to indicate their preferences or interests with respect to data sources recommended by the application developer. One of these declarable preferences is the expression of satisfaction about what they are informed of or about the manner in which they want to be informed, for example, via electronic mail or immediate presentation.
Application Event Distributor/Transporter
This is responsible for connecting the middleware system with the user side application so that both sides can be supplied with awareness information. Any awareness object that captures users' actions or data changes can activate this process. It has the role of receiving the signals from awareness data objects and parsing them to the event handler in the middleware. This component then sends the data to other layers or to a process that compares and evaluates the information to find out what should happen next.

Example:
The application database had two base data tables user and office. In addition, an awareness data class presence was created. In the application, one member moves from one office to another. This change of a user's working place may be captured by the application and raised as a query to the data storage to change the related data, awareness data presence. This activity should be also streamed to the event handler middleware system so that other users also know about the office change of the member.

These modules support the process of the application designers formulating specific needs for visualization using the database model. Their intent is not providing a visual programming language for constructing a application data model.

3.3.2 Cache Handler
In a visual database exploration application, the visualization is simply representing the database query result. Thus, visual database exploration usually starts from a database subset [LG95]. However there might be applications that just read out the whole data source and want to show it, if possible, all at once. It is highly dependant on the application designer's concept for the visualization and the subset size. Thus, it is necessary to have an effective cache support to prepare the fast data supply and also to handle the storage of the data on the application side until it is permanently saved in the database.

Data Preparation Handler
Instead of responding to the query that asks for more data supply, it is desirable to prepare data sets beforehand to accelerate the data feed. Knowing which data is interesting for the visualization, it is possible to prepare the data beforehand. For example, the data that is needed in two minutes could be prepared while the process is not busy and before a data request is sent from the application side. This could shorten the response time for the application and thus the visualization time. This case was implemented in section 5.4.7. Since the data source should be read out for a time interval of one hour, for example, and the data should be supplied after
the time attribute, a process is made to prepare the data as it is requested from the visualization and stored on the cache. In so doing, it was just an unavoidable fact that it was required to supply up to 128,000 objects per query in the maximum case.

**Parallel Access Handler**

In case the application is a collection of processes, it is highly demanded to have a cache of the application data that can be accessed by different processes. For example, one process constantly reads out the data from the database and saves it on the cache. Another constantly serves out of the cache to visualize the data. Thus, it is important to have a cache system handling the data in a shared memory area. This is a big advantage when providing large data sets into an application that is supported by several data manipulating processes. For the case of when the application is implemented by several developers, it can also provide flexibility for their work in supporting modular programming.

### 3.3.3 Communication Channel Handler

The role of the communication layer is to connect all components of the interaction model. The communications going from one component to the other have to be managed without analyzing the content of the information. The communications
can be as complex as a network system: from a user to another user, from a user to the database, from one layer to another layer, from the database to all users, etc. To support all varieties of communications available, the communication channel handler provides a set of fundamental roads. The communications are limited to three types of communications: among users, to the database and among application domains as shown in the figure 3.4.

**User-to-User Mediator**

Not only in a data visualization environment but also in a general application environment, it is important that the communication between users is supported. This will stimulate all kinds of collaboration. In doing so, the identity of the users is important in transferring the information among them. Let us say a user in an environment wants to send a small message to another member who is shown as present in the application. For this, a fixed form of an object is used.

*Example:*

A user $CL_1$ wants to send a message to all the other users working in the same environment and connected at the moment and sends an object as follows:

```plaintext
// DEFINE AN OBJECT PACKAGE FOR COMMUNICATION
A package is:
    senders_identity
    receivers_identity
    message content object
    distribution_type
end package

// HANDLE THE MESSAGE DISTRIBUTION TYPE
if distribution_type = to all user then
    while users to be inform exists
        get a user’s identity
        assign the object to the package
        send the package
    end if

if distribution_type = to a certain user
    get the user’s identity
    assign the object package
    send the package
end if
```

As the value of the attribute `distribution_type` of the message, for example `shout`, is used in a case study, signifying that the message object should be sent to everybody who is present in the environment.
Database Mediator

In a visualization application environment, different communications to the database can be made. Especially diverse queries including one caused by user interactions or application inner processes can be sent through the communication channel to the database. The database mediator should also synchronize these queries, so that the queries are sent to the database in a reasonable rate, and it should also return the result to the right sender. In this thesis, this database mediator is designed in two separate roles. One is responsible only for sending queries and receiving the result of the database. It does not care about the content. It checks only that the result is not empty and that the query process is working well without being interrupted. The second process takes care of the results, thus it can be more application dependently implemented. In case the communication goes to the database, the identity of the database is ignored using the package object above, i.e. an empty identity. However, if there is more than one database to handle, then the identity of the database might be important. Such an example can be found in the project “A Tool Set for the Virtual AEC Company (ICCS)” project [Too] and is called “Web Top”. The ICCS project concerned the field of database systems, Internet technology, and information visualization to provide a system that can enhance collaboration among groups through the improvement of data visualization.

Application Domain Mediator

In general, visualization applications are made to be used on different system domains and platforms. On the one hand, it is difficult to implement them using only one language or one platform. For example, one programming language allows providing the application only on specific hard and software environments and another allows providing on an Internet-based environment. Thus, it is desirable to give a middleware system that supports the communication among these different platforms to exchange data. This might be difficult to realize. On the other hand though, it encourages showing the same data using different concepts of visualization on diverse environments at the same time.

Some experiences visualizing the same data source in different ways were done in an ICCS project. In the third case study of this thesis, section 5.4, a connection was enabled between a relatively isolated environment to a web-based application to exchange the event on each side independently of how the information is visualized on the screen.

3.3.4 Comparison with other conceptually related works

The problematic of poor integration between database techniques and visualization techniques are recognized by many researchers of different kinds of visualization including 3D graphical applications [Owo02] and scientific visualizations with weather-related data [Tre00]. Human Computer Interaction (HCI) shares the same opinion
on this topic and Cruz proposed a combined research effort in HCI and information systems [Cru96].

It was also remarkable that the integration between the database and visualization techniques was not sufficient for applications, and this thesis proposed necessary measures to address this problem in the framework. The architecture of the framework outlines the fundamental bases in extendable layers of modules according to specific requirements. Finally, it provides guidance for development platforms or environments for visualization applications. The challenges encountered include supporting several visualization-specific data definitions and transformations, connections among interaction components, data adaptation possibilities of the user’s preference and high-speed data supply for applications. It differs from many toolkits in that it does not provide a class or library of pre-defined visualization primitives or an automated generation of interface using visual primitives. In so doing, those pre-defined primitives fail to provide the right support in the context of visualization, because they are too limited to support the concept of awareness and aesthetic visualization. With the forthcoming database framework, the application designer can expect that they have full access to the database system including the metamodel. This framework brings both sides, the application designer and the user of the database, tightly together with full respect of the designer’s vision for visualization.

Similar to the concept of this work, Teallach [GCK+98] aims to provide a rich support for application designers to achieve their work in the database context. Using their presentation model, the designer can describe the relevant objects to the visualization in relation to the user interaction. This is a database user interface, which might be compared to the modeling layer of the forthcoming database in a much more structured way. It might be an helpful system for working with a large size schema of the existing database so that the application designer could describe which of those objects might be interacted by the user and how, etc. However, it was difficult to be convinced whether the work would be worthwhile for a small size database schema, making libraries of the interaction objects for the application designer instead of opening the access to the database through a more simple interface, such as showing and allowing them to model on it.

The concept of Rivet [BST+00] is to visualize computer systems where all operations are allowed without quitting the system Rivet, from data transformation to the table to the visualization using primitives. It has powerful transformation functionality and the idea to provide everything in a coherent system is an advantage. However, since it is a computer system visualization environment, the potential of the visualization concept side is underestimated.

The communication system of the framework is a widely used concept. It can
be also used for distributed databases creating more than one database mediator of the system and starting them on local machines. They can then communicate with each other such as several local database servers (see Web Top server, in [Too], documentation → ICC System Aspects → Component Architecture). In this work, developing a communication architecture based on one central database system was the main topic. The question of how many database mediators can be created is left open to the application designer’s decision. For example, more than one database mediator was built in the third case study. As a result, it was possible to have better performance supplying large data sets to the data visualization application. Similarly, it was reported that such communication architecture is capable of supporting efficient data movement among heterogeneous hosts and can be scaled for large data sets in a system for simulation and visualization [vHd98].

In terms of the software system design process and persistent database, a more database-centric approach was developed by Kobler. The eXtreme Design Approach is a help for the design process, especially for component and information modeling, with regard to the evolution process of a system and user involvement [Kob01]. It is strongly based on object-oriented paradigms and uses an OM model [Nor93, Nor95] to define its algebra. Based on its characteristics including support for design process, information modeling and database-centric architecture, this system shows a great potential to be extended for visualization applications as well.

3.4 Properties of the Application Framework

The main characteristics of the application framework including its data centric nature and how it is forthcoming to the interactions in application development and management are enumerated.

3.4.1 Interaction Data State Example

In the figure 3.5, an example of the data states using the framework by means of an application that visualizes a subset of a database on the interface can be followed. At the beginning, a part of the data from the database is derived, i.e. application data. Next, these data structures are reformed or used to build new data objects as defined in the data modeling components layer. These might be created just for the visualization and/or serve to enhance the awareness of the user through event or awareness information. This reformed data is saved in the cache until the application side process accesses and uses it for the visualization. At the same time, there are some running processes in the middleware. For example, a process controls whether there is enough data saved in the cache. If not, it sends queries to the database and prepares the data for the cache. This can be caused by time or the amount of data in the cache. Using the application, the users could also have the possibility of influencing the awareness data according to their preferences.
Also, some awareness information might arrive from the database to the application data cache to be sent to a certain person or all users. Using this framework, the creation and handling of awareness data should be made easier and the work of an application developer should be reduced. Furthermore, it should allow an enhancement of the users’ and application designers’ control over these data sources.

**Figure 3.5: Example of the process using the framework**

### 3.4.2 Collaboration Model

In making the framework for the forthcoming database, a collaboration model is built to clarify how the collaboration among user, database, and application designer should ideally look like. It is shown in the figure 3.6. From the user’s viewpoint, a person who interacts with the data represented in a visualization application should have the possibility to express her interest to affect the system. From the application designer’s viewpoint, the data can be freely remodeled and configurable for awareness information. And the database informs the user about the data new states, provides functionalities to form application data and supports the application designer’s work by offering transparent and higher level database access.
As the model shows, the collaboration should base on and function for the transparency of data. It can only be guaranteed by well-functioning communication paths and flexible and adaptable data modeling by the application designer. It should also include the users’ preferences in the application environment.

![Collaboration Model](image)

**Figure 3.6: Collaboration Model**

### 3.4.3 Characteristics of the Application Framework

This thesis proposes a framework, which outlines the necessities for visualization application and it is powerful in binding the database role with visualization applications.

- The concept allows high level data modeling of the application objects and its manipulation from the visualization side.

- In the development of an Internet-based groupware application, the concept puts the awareness data in the center of the collaboration environment, as it is important for user interaction. The application developers have enough elbowroom to define what is awareness data in their application and to adapt to it.

- Data access is allowed for both categories of actors: users and application developers in different levels. This postulates the necessity for communication and awareness of each other’s interests and actions. This also helps maintain the application as it evolves, thus, reducing the designer’s work.

- The middleware system is forthcoming to the designer and the user with respect to awareness data handling and declaration.
• It is modular and extendable to different application domains using different languages.

• It stimulates more communication and collaborations (see figure 3.6) among the surrounding data sources.

3.4.4 Users of this Framework

Three kinds of users can benefit from the framework. The first one is the database expert. It is very common that a database expert works together with other developers and supports the data management services. In building an application together, the framework could serve as an implementation platform for the database expert. Using the framework, the database expert could provide more fancy applications to the application designer, especially to spare him work. The application designer might be the second possible user of the framework. The framework can be further adapted to the specific visual concept of the application designer and for this, she can use all of the functionalities offered by the framework. Even though the application designer decides whether or how much the users can access the application data, the users are the third user class of the system. The framework can offer them a functionality to model their preferences using a database schema and in some cases, the user of a visualization application will positively benefit from the framework functionalities by receiving messages from the database without having done anything.

3.5 Recapitulation

To make a useful visualization application, effective data handling systems are needed in addition to an interactive graphical user interface. Users’ activities on data sources that have to be represented in such an environment are a strong motivation to require a flexible and specific application framework. Also the application designers’ work in handling and adapting the data to application data affects their work process and disturbs their concentration on the visualization. Especially the use of data sources for enhancing awareness according to interactions and events in the application is very crucial in developing a sensible and reasonable collaboration work environment. For this purpose, an application framework focusing on the effective use of data designed for the application is suggested.

The framework stresses the interactive processes in handling data and especially the role of the database. It is useful during and after the development of data visualization and motivates the process of using and maintaining the application after its development. The collaboration model describes the effect of the concept. It puts database functionality in the center of the collaboration, proposes users data access for personalization and assures connections among actors and data sources. The
practical position of the framework is between the database and the application as a middleware system containing three main layers with different functionalities and inner layers. With this structure, the systems imbedding the framework encourages and supports forthcoming behaviors in a database in the context of a visualization concept.

Out of this application framework, a Java application middleware system is implemented. This can be extended to an Internet-based application and allows different domains to interact with each other in different work environments. This Java-based system will be introduced in the next chapter.
Chapter 4

AwareDB: Java-based Application Middleware System

This chapter provides a description of the prototype implementation called AwareDB. AwareDB is a representative model of the forthcoming database framework whose functionalities have been developed with the test cycles described in the next sections 5.2, 5.3 and 5.4. AwareDB’s architecture allows it to be was successively extended using modules. This prototype helped us to establish an effective basis for dialog with the users, test and evaluate the framework, and receive feedbacks on the prototype and concept. Thus, it fulfills many of the design ideas postulated for the application developer. The process of prototype development was user-centered design, and, as result, it brought out a vertical prototype [GLZ99] that contains important high and low-level functionalities of the forthcoming database concept.

The AwareDB served as an application framework for three case studies. Through the case studies, the AwareDB was optimized in many aspects and sufficiently proven according to the concept proposed in this thesis and its implementation. Strong characteristics and a diversity of the visualization applications motivated us to enhance AwareDB’s adaptability. However, its technical requirements sometimes forced us to make more than one version, costing much reimplementation work. Even though a Java-based framework implementation has already existed, most of the system had to be reimplemented in C++ due to the change of the case study environment. However, this work paid back through the broadening of the flexibility and adaptability of the thesis concept.

In this chapter, the AwareDB implementation is outlined. First, AwareDB design choices are described. Then, the elements of the prototype are presented showing the interfaces and examples. The example data is drawn from the case studies and uses mostly Java programming language notation shown in plain text or pseudo code shown in boxes. Finally, issues that came up during implementation are discussed.
4.1 Design Choices

The requirements of the case studies with regard to design choice can be shortly summarized as follows:

- The visualization logic has support has to be maximized, and it should provide a base for different kinds of visualization.
- Access to the World Wide Web is required.
- Fast implementation and reimplementation possibilities are asked for.
- Implementation for additional functionalities should be easily incorporated into the system.

Facing these requirements, the use of the client/server architecture [Lew98] is basically taken to provide the prototype of the concept. AwareDB is a middleware that serves as a framework for communication between the client and server portions of a system and contains also the functionality as database middleware [MG00, Emm00]. Fat-servers are known for easy manageability and the fat-client system is more popular because of its programming ease. The latter tendency could be also observed in the applications considered, because the persons responsible for the visualization concept were mostly client programmers. To give a good balance, it was decided to move the functionalities to the middleware side and plant as many functionalities as possible in each layer similar to the component based n-tier model [Ste98].

Furthermore, a 3-tier client/server system was chosen. Keeping the idea of a traditional 3-tier model, the second tier where the work of the application designers and AwareDB meet together was divided into two parts.

<table>
<thead>
<tr>
<th>Tier 1</th>
<th>Visual logic of the application, handled by a client system. The processes handle purely the mapping of the data and visualization graphics, navigation, and interaction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier 2</td>
<td>Processes related to the visualization and needed to be run on the server. This includes some calculations, or interaction manipulations of the visualization concept.</td>
</tr>
<tr>
<td>Tier 3</td>
<td>Database connection and transaction management</td>
</tr>
</tbody>
</table>

Java was chosen as the implementation language. It was necessary to provide access through the Internet and many users were using different platforms. Deciding to use Java as the implementation language (Java2 Software Development Kit, v1.4.1) and providing all inner system connectivity, AwareDB became a web-based computing
framework [KR03]. At the time of this study, the Java frameworks had not yet developed to the point of J2EE. However, the use of a Java core system enabled us to move the framework to the web-based computing framework and provide the application designers more freedom in realizing their concepts. During the last stage of the work, the framework was implemented using C++. This was conceived as an extension of the framework, because it could be connected to the original AwareDB. However, due to the entire system domain that was defined by the other technical factors in the project, the functionalities were implemented in C++. If the Java language had been used, it might have been impossible to respond to the high speed requirements (it is a well-known fact that Java’s performance suffers with speed) [vRS02].

Considering that the most commonly used database system in the collaborating group was MySQL and also the application designers were familiar with it, MySQL was taken as the base database management system. In addition, there was Oracle, which is also a relational database management system (RDBMS) and used in several building applications in advance. For implementing the prototype AwareDB, using both databases did not make much difference, since a small part of their core functionalities were used. Most important was that the application designer already felt familiar with MySQL, and it also offered high compatibility with several scripting languages that are used for application designers’ work. Also, JDBC API simplified the database connections [Cor98, Rit98]. Having one central database on the server side, the management of data was made easier, the database connections were simplified, and maintainability was enhanced.

### 4.2 Data Modeling Components Layer

As components belonging to the Data Modeling Components Layer in the framework, functionalities to make the database schema transparent to the application designers and partially to the users were mainly implemented. In addition, application object modeling possibilities were offered. Providing this access facility, the application designer was enabled to model application objects according to the visualization concept, upholding the connection to the original database schema. Also, in some cases the users had access to the modeling results of application designers and were able to express their preferences as their interest.

All three case studies were based on a relational database, Oracle or MySQL. For the first two case studies, these functionalities were provided at the same level as the user interface. It was linked as an HTML reference to the user interface and used PHP and Java Script for the graphical representation. This graphical representation is independent of the application made by the application designer.
4.2.1 Visible Database

The functionality of the interface in figure 4.1 is to note existing tables in the
database and designate which ones should be transparent to the user side, to fi-
nally offer a possibility of directly manipulating their interests. Assuming that the
visualization application is already connected to a database, the interface showing
the database schema can be opened choosing an HTML link. In case the visualiza-
tion application has not yet started, this functionality can be started separately.

![Graphical interface for making tables visible or invisible.](image)

Figure 4.1: Graphical interface for making tables visible or invisible. It shows tables
read out from a course application database. Especially, the attribute *thedeate* of the
table *works* is processed as visible on the figure.

Using a similar interface, the application designer can define which tables should be
candidate tables by assigning an attribute *visible*. In this interface, such visible ta-
bles of the application database are shown first, displaying tables from the database
schema with additional information about whether they should be visible or not. If
a table is defined as a visible one, it will be accessible by users through an interface
to express their interest on certain information, see figure 4.3. Choosing a table
enables an administrator to get all relevant attributes. The application designer
then selects or deselects to make some attributes visible or invisible to the users.

The APIs used for setting and reading out the visible tables are as follows:

- void setVisible(tablename, Attr)
  It makes an attribute *Attr* of a table *tablename* visible to the users.
Hashtable getVisible()
It retrieves all tables that are visible and returns those in a Hashtable back.

These values can be changed at any time by persons who are recognized as administrators or application designers by the system. In the case studies, there was a table admin containing information on the administrators.

Example:

There are tables for a lecture environment including groups, people, works, subwork, comment, lastaction etc. Each of these tables has its own attributes. In case of the table works, the administrator of this lecture environment decided to show only the attributes creator, exercise and concept. This can happen either through the graphical interface or the API. In the latter case, a Java Vector with values of the attributes should be prepared as follows:

```java
Vector attrs = new Vector();
attrs.addElement("creator");
attrs.addElement("exercise");
attrs.addElement("concept");
```

and then the API setVisible() is called.

```java
void setVisible("works", attrs);
```

As the example shows us, the middleware should transform all the retrieved data from the relational database into the Java object-based environment.

4.2.2 Interest Setting

With the given visible tables, administrator and users can say what information they are interested in to show or to receive. The distinction between the administrators and application designers are done by the definition of user identity attributes. In doing so, the person who is allowed to use this interface has to be controlled from the application side. In the case studies, certain users were defined as administrators and only their interfaces had an additional link to the functionality to set interest. The term application designer is used as higher definition of administrator in this text.

AwareDB clearly separate two possible interfaces. Figure 4.2 shows how the application designer can work with the functionality. There are three possible cases: insert, update and delete, which means a transaction from the application to the database. For each of these cases, there are three ways to define how the information has to be considered. In case of inform all, the database will send the information to everybody simultaneously. If refer to personal setting is chosen, the
transaction will trigger a data transfer to only the users who are interested in it. Otherwise, if the application designer selected *do nothing*, nothing happens. The

![Graphical interface for setting application designers interest using the visible tables of the database. The interface shows that the application designer wants that all users get informed, in case a update transaction with respect to the value of the attribute *exercise* of the table *works* is evoked.](image)

normal users can express their interest on certain data in two ways, interested (*inform me*), or not interested (*do nothing*), see figure 4.3. The interest settings of the users can influence the designer’s decision to modify the behavior of AwareDB. Ultimately, the designer can determine how information is distributed in a system (i.e. users can give their preference, which can affect the designer’s decision). To set the interest of users and application designers, the following methods are used:

- **void setInterest(tableName, transaction, doAction)**
  An application designer wants to activate the action *doAction* in case a transaction occurs on the table *tablename* in the system.

- **void setInterest(tableName, transaction, interested)**
  Instead of selecting *doAction*, possible actions could be defined by numbers, *interested*. Accordingly, actions to the interested users are implemented by the application designer.

The transactions and the actions could be individually defined with regard to the applications. In the case study Virtual Office, section 5.3, different adaptations to provide more options for how users can be informed and the functionality to create an additional attribute that stands purely on the application side were shown, see figure 5.13 and figure 5.14. Also, the same method with parameters of different data
types was implemented.

Example:

A user with the unique identity $U1$ is interested in the data type works. She especially wants to be informed in case a data element of table works is inserted in the database. This can be arranged by the interface when the input value inform me is chosen for the action insert. To define this using API, setInterest should be called.

```java
void setInterest("works", "insert", "inform me");
```

Figure 4.3: Graphical interface for setting user interest. Compared to the interface above (figure 4.2), it shows the limited possibility of a user in setting their interest, either by saying inform me or not.

Currently, AwareDB allows one to determine only what the user is interested in, not uninteresting information. Therefore, there can be no conflict by defining the interest of each user with another user’s interest. However, in allowing more complicated choices to the users, it is recognized that there is a great danger that the user can get overloaded with too much information. Instead of letting the user say what she is not interested for, AwareDB can filter information by comparing the user’s interest with interests of others at a certain level.

/user A’s formulation/: if Work W’s attribute updateDate changes, then inform me.

/user B’s formulation/: if Work W gets updated by anyone, then inform all users.
This kind of situation will happen quite often during a collaboration. To protect user A’s private interest area, AwareDB would make user B’s rule smaller in that it changes *all* to (*all* − *userA*). However, the decision whose interest has higher priority might differ depending on the application.

### 4.2.3 Comparison between Concept and User Interest

The designer’s decision about what to inform and how to inform can turn out to be inefficient after a while. The designer had defined that messages about a change should be delivered to everybody, but it might not have gained as much of the users’ interest as expected. To make the designer conscious about the users’ up-to-date interests in their working environment, AwareDB can show discrepancies between the designer’s settings and users’ preferences. Seeing these values, the designer can adapt the behavior of the application system step-by-step to match users’ tastes. Figure 4.4 shows the view half an hour after having set this functionality in the application.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Action</th>
<th>Setting</th>
<th>User nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>computers</td>
<td>insert</td>
<td>do nothing</td>
<td>0/41</td>
</tr>
<tr>
<td></td>
<td>mobile</td>
<td>inform all user</td>
<td>1/41</td>
</tr>
<tr>
<td></td>
<td>location</td>
<td>inform all user</td>
<td>1/41</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>inform all user</td>
<td>1/41</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>do nothing</td>
<td>0/41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Action</th>
<th>Setting</th>
<th>User nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>vacations</td>
<td>insert</td>
<td>inform all user</td>
<td>0/41</td>
</tr>
<tr>
<td></td>
<td>update</td>
<td>do nothing</td>
<td>0/41</td>
</tr>
<tr>
<td></td>
<td>actYear</td>
<td>inform all user</td>
<td>1/41</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>do nothing</td>
<td>0/41</td>
</tr>
</tbody>
</table>

Figure 4.4: Comparison of settings among users and application designer. It shows a cutout of an evaluation site. The comparison related to two data tables, *computers* and *vacations*, are shown. The yellow color is chosen to express the table, which is updated last. In the table in yellow, the data type *vacations*, possible transactions, and interest settings designated by the application designer are shown. The last column shows how many users are interested in getting the information the designer defines as system behavior.

The number of users who declared themselves interested in getting messages about certain actions, including the insertion, deletion and update of attributes will be
shown as a number in the user nr column. For example, at this very moment, one of 41 application users has set her interest about the update of attribute actYear. The yellow background color means that the table vacation was updated by a user recently.

4.2.4 Information Distribution

The evaluation of users’ interests happens through the right distribution of the information as declared by the application designer and users if an event in the application environment or in the database is recognized. The event is parsed to a data type MAction that is only used in the Java side of AwareDB and mainly handled on the cache while the application is used.

```java
public MAction(Vector param) throws ModuleException {
    this.actor = U1;
    this.action = MAction.INSERT;
    this.attr = null;
    this.type = "works";
}
```

It contains variables describing who is the actor and what kind of action is evoked by the event. If an event on the application side occurs which might match a user’s interest, AwareDB goes through the existing user interests and searches one that matches the event. To then distribute the information as expected, AwareDB creates the result of the evaluation as an object that can be finally printed out as a sentence. This object is called AwareInfo and the structure appears as follows:

```java
AwareInfo() {
    action String;
    onType String;
    condition String;
    value String;
}
```

AwareDB finds the user who has to be informed based on the interest information. Furthermore, it determines whether this person is connected to the application or not by using the connection information including IP, user id, etc. Once the AwareInfo arrives on the application side, the application can use a static method AwareInfo.write() to print it out as string text, which is the most simple form. All the variable values of the object AwareInfo can be listed, invoking get* methods.

4.2.5 Meta Tables and API

Due to the additional data structure needed to handle awareness information including users’ actions and events, AwareDB stores its tables as meta tables in the existing application database: metaVisible, metaInterest and metaAdmin. In a DBMS
The three data tables were used mainly for the AwareDB processes in which users’ interactions with data are permanently stored. The `metaVisTable` stores the result of the interaction given in section 4.2.1 and `metaInterest`, the result of the interaction given in section 4.2.2. AwareDB handles the user’s roles independently from the application. That means an administrator in the application environment is not necessarily an administrator in AwareDB for the awareness data modeling and management. The application designer can define more than one administrator in AwareDB, which is stored in the table `metaAdmin`. A user can be an admin or member in several environments, and an environment can have more than one administrator. This table is not dependent on application data about users. A part of AwareDB’s API `aware.meta` has methods to convey the transaction to these meta tables including `setInterest`, `setVisible` and `setAdmin`. To read out the values from the above tables, there are methods called `get*` for each table.

### 4.3 Cache Components Layer

The Cache Components Layer was especially used for the transformation aim and to save data for a short period of time before it is feed to the visualization scene. All data retrieved from the database arrived first in this layer and was sometimes saved in the cache. The case study, in which the urgent necessity of the cache functionality was recognized, involved the application of a 3D immersive VR application. The reason was the amount of the data that had to be supplied in a short time frame and the speed required. Thus, it will be described here how the parallel access to the data was realized and how the high-speed query was answered by preparing data sets in the cache. The study was done using the C++ language and pseudo code is used here to describe the implementation.
4.3.1 Parallel Access in Shared Memory

The C++ notation `pfMalloc` was used to reserve the memory for all the possible data sets beforehand.

<table>
<thead>
<tr>
<th>Initialize shared memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>// ALLOCATE THE MEMORY</td>
</tr>
<tr>
<td>set *arena;</td>
</tr>
<tr>
<td>// FIND FREE MEMORY</td>
</tr>
<tr>
<td>set arena to pfGetSharedArena();</td>
</tr>
<tr>
<td>// ALLOCATE MEMORY FOR APPLICATION OBJECTS</td>
</tr>
<tr>
<td>pointer_to_first_object</td>
</tr>
<tr>
<td>= (application_object_size*) pfMalloc(max_length_of_cache * sizeof(application_object), arena)</td>
</tr>
<tr>
<td>// INITIALIZE THE SHARED MEMORY CREATING A LINKED LIST OF POINTERS</td>
</tr>
<tr>
<td>for each of the shared memory place</td>
</tr>
<tr>
<td>do put a pointer</td>
</tr>
<tr>
<td>do initialize its previous and next pointer</td>
</tr>
<tr>
<td>end loop</td>
</tr>
</tbody>
</table>

It is important to reserve this kind of shared memory `arena` in C++ to make it accessible by both the application side and AwareDB side processes. In this reserved memory area, AwareDB could store (write) the newly retrieved data sets, and the visualization processes could be read out. Both processes were managed in parallel so that the application side never could notice when the data sets were retrieved from the database. The AwareDB side process was realized as a child process created using the C++ notation `fork()`.

4.3.2 Preparing Data Sets

In contrast to MOVE [GMP+02] in which shared artifacts are used, the system performance seems to be much related to the middleware. The reason might be the amount of the data supplied in a visualization time frame and how fast the visualization time frame changes, see section 3 in [Moe04b]. In this case, the time frame played an important role for the visualization, for which the AwareDB functionality, preparing data sets, was implemented. All the data sets in the database had a time stamp such as date, year, time, etc. In case there was no time stamp for the data sets, the application designer added a kind of time stamp attribute. Additionally, the visualization side had their own time frame depending of the visualization granularity. The decision how to tune the time frame was done by the application
designer. For example, all data sets of a whole day could be rendered on the visualization side in a few time frames. Thus, the method was implemented so that the application designer can set the desired time frame before the application starts. To support this time frame related data request with high-speed performance, data sets were prepared beforehand. In the next code, the time stamp of the data sets was used as a control value for retrieving the data.

<table>
<thead>
<tr>
<th>Prepare Data before the data request</th>
</tr>
</thead>
<tbody>
<tr>
<td>// PREPARED DATA COUNTER</td>
</tr>
<tr>
<td>set prepared to 0</td>
</tr>
<tr>
<td>// TIME INTERVAL FOR THE DATA TO RETRIEVE AND PREPARE</td>
</tr>
<tr>
<td>get frame_unit</td>
</tr>
<tr>
<td>// DATABASE QUERY SHOULD BE SENT call_count TIMES</td>
</tr>
<tr>
<td>get call_count</td>
</tr>
<tr>
<td>// DEFINE THE POSITION IN THE CACHE</td>
</tr>
<tr>
<td>get last_retrieved_object_in_cache</td>
</tr>
<tr>
<td>// PREPARE DATA FIRST TIME</td>
</tr>
<tr>
<td>if last_retrieved_data_object_in_cache = null then</td>
</tr>
<tr>
<td>set position as pointer of the last memory in the cache</td>
</tr>
<tr>
<td>set first_retrieval to 1</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>while database is not empty and prepared &lt;= call_count do</td>
</tr>
<tr>
<td>// MAKE THE QUERY CONTAINING NEW CONDITION</td>
</tr>
<tr>
<td>set time_stamp = + frame_unit</td>
</tr>
<tr>
<td>formulate new_query_condition replacing time_stamp;</td>
</tr>
<tr>
<td>// RETRIEVE DATA AND STORE IT IN THE CACHE</td>
</tr>
<tr>
<td>retrieve data sets with new_query_condition</td>
</tr>
<tr>
<td>if (!retrieved data sets = null) then</td>
</tr>
<tr>
<td>prepared++</td>
</tr>
<tr>
<td>put in the cache after the last_retrieved_object_in_cache</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end while</td>
</tr>
</tbody>
</table>

The variable \textit{frame\_unit} is to be set by the application designer, because these are dependant on the time frame in the visualization and the speed performance. The variable \textit{call\_count} is tuned in observing the cache use and frequency of the data request from the application side.
4.3.3 Interpreted Data

In the third case study, see section 5.4.6, the handling of interpreted data was required. A meta data pool was made in the middleware to keep the information in a container (list) and manage this specific data object class as a normal data object.

<table>
<thead>
<tr>
<th>Build a container of Interpreted Data objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>//DEFINE A INT NUMBER FOR THE NEW ENTITY</td>
</tr>
<tr>
<td>set entity_index</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>// MAKE A LIST TO CONTAIN ALL INTERPRETED DATA CLASS NAMES</td>
</tr>
<tr>
<td>set itp_entities as empty at the beginning</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>// PREPARE A STRUCTURE FOR THE INTERPRETED DATA</td>
</tr>
<tr>
<td>make a container class information of an object class.</td>
</tr>
<tr>
<td>set container size</td>
</tr>
<tr>
<td>for each interpreted data object class</td>
</tr>
<tr>
<td>set entity_index</td>
</tr>
<tr>
<td>increase entity_index</td>
</tr>
<tr>
<td>reserve a vector to add other information belongs to the object</td>
</tr>
<tr>
<td>end loop</td>
</tr>
</tbody>
</table>

This Itp data (interpreted data) class includes many other methods to supply the object to the visualization and manage the process correctly. In this work, the Itp data related information is not stored to the permanent database, but this can be done when the programmer prefers it. Each object of the interpreted data contains information, such as other normal data, object name, id, etc. One conceptual difference compared to the normal case is the attribute. It should have an empty attribute vector, because it is originally an attribute itself.

The interpreted data object is defined using `setItp(Name, Parent, Attribute Name)` and the value of each object is first set calling the method `get(Name, Value).` Thus, the `get`-method is actually creating each object. There were two choices in this process. First, the real number of the data object was created. For example, `setItp(“finance”, “department”, “budget”);` in which the budget value is 120,000. Calling the method `get(“finance”, 4)` should ideally have created 30,000 objects which marked as being interpreted data. As a result, the system would no longer have been able to supply the visualization data with the requested rapidity, due to the data object creation cost. Thus, the second choice was taken; only one representative object having the number of objects 30,000 was made as a value of an attribute. In this way, the data sets were supplied without high calculation costs.
4.4 Communication Components Layer

The communications in AwareDB are based on sockets. This is founded on the client-server paradigm where the client initiates the peer-to-peer connection and the server waits for incoming communication requests for certain jobs [CS94]. The socket connection between the user and the AwareDB stays open for a session until the user disconnects by exiting the application. The connection between the database and AwareDB stays as long as the database mediator is not turned off. Next, the Package is introduced, the transfer unit going through the connections, and then the details of each connection in AwareDB are described.

4.4.1 Information Transfer Protocol - Package

For data transport, a Java object Package is implemented. It contains information about the sender, receiver, and content of a transfer. Using the information in the package, the information can be redirected from one user to another. In addition, the communication components layer contains mechanisms whose duty is to enable the communication between clients and the database service center.

```java
public Package(IDCard to, IDCard redirect,
               Object content, IDCard from) {
    this.to = to;
    this.from = from;
    this.relay = redirect;
    this.content = content;
}
```

IDCard is a client description containing values about the user’s identity and host name, etc., and from, to and relay describe who sends the package with content to whom and to whom the message has to be redirected by AwareDB. Thus, they are all functions of user identities IDCard. The content is an object which can be any type of the allowed Java data types.

**Example**

A client with identity C1 wants to send client C2 a message. Then, the Package carrying the following message is created:

```java
new Package(C2_IDCard, C1_IDCard, null, message)
```

A client C1 sends a query to the system and the result should be sent to the user C2. Then, the Package carrying the following message is created:

```java
new Package(DBService, C1_IDCard, C2_IDCard, message)
```

As it is shown in the above example, objects such as messages or database queries can be sent to the system processes such as DBService.
4.4.2 Client’s Connection

In figure 3.4 of chapter 3, the main communication paths in AwareDB were shown. How the system and users are connected is shown more in detail in figure 4.5.

Figure 4.5: On the left, the users’ connection to the framework system AwareDB, which is a part of the figure 3.4 of chapter 3, is highlighted in red. On the right, the detail view of the highlighted part is shown. For each user of the application, a socket to the system is opened and kept open for a session. The socket thread handler on the framework side is responsible for managing the synchronized communication arriving through the sockets. It also controls the information distribution to the appropriate client and communications among users.

A user’s connection to AwareDB is applied by calling one of the following methods:

```java
void connect(String user_identity);
void connect(String name, String password);
```

A `user_identity` can be a `java.lang.String` value. It is a name such as “Paul” or a `java.lang.Integer` value such as “1” wrapped as a `java.lang.String` value. Important is that the application side has to continually have a unique identity for each user during a session. In case the application designer wishes that AwareDB takes control at the very moment the user connects to the environment, the tables containing user information must be arranged so that both AwareDB and the user application understand each other. To connect and disconnect from AwareDB, the commands `connect(…)` and `disconnect(…)` should be used. There exists a service running on the AwareDB side. This is already connected to a database and waits for user connection requests. This is an instance of Java Thread and a socket handler. It notes the location of the port in a web accessible file system at the moment of creation.

On the one hand, this thread handler is waiting for a request from any client that can send and receive the package as defined in AwareDB. On the other hand, when a client connects to AwareDB, it automatically creates a socket and asks the server side to answer, see figure 4.5. If this socket connection between the client and the
socket handler on the server side gets established, communication can begin. The server side thread handler creates a `java.lang.Thread` instance for each user’s socket request. So, there is a socket thread responsible for the communication channel to each user.

### 4.4.3 Communication to Appropriate Database

The integration of AwareDB with the data visualizing application begins with the connection to the application database whose database schema represents the application data environment. It is called *Database Mediator* in AwareDB. Designers prefer different kinds of databases depending on their experience. AwareDB offers a class `DBConnection`, which can be easily extended to build a connection to the one the designer has chosen for the application. The following example shows the connection to an Oracle Database through a JDBC connection.

```java
public class CertainDatabaseConnection extends DBConnection {
    public CertainDatabaseConnection(String driver, String user, String pwd, String connect) throws SQLException {
        super(user, pwd, null, null);

        // describe the database that is used for the application
        this.database = "jdbc:oracle:" + driver + ":" + connect;
        try {
            // describe the driver
            Class.forName("oracle.jdbc.driver.OracleDriver");
            catch (ClassNotFoundException e) {
                System.out.println("Could not load the driver");
                e.printStackTrace();
            }

            // set connection
            this.setConnection(DriverManager.getConnection(this.database, this.username(), this.password()));
            this.connection().setAutoCommit(false);
            this.printWarnings();
        }
    }
}
```

Once the connection to a database is established, AwareDB reads out what kind of data structure is used in the application. It creates some system information such as
some event-controlling program and saves it to the database. Until now, AwareDB connections for databases Oracle and MySQL have been implemented and tested. All the queries inside of the system to the database service manager Database Mediator are packed in a Package, the communication transfer format in AwareDB.

![Diagram of middleware framework and client applet](image)

Figure 4.6: On the left side, the framework and its Java class ClientApplet are shown. The visualization applications need to create a class that extends this ClientApplet class to dock to the framework, as shown on the right side. Consequently, the control of the communications is taken over by the framework.

### 4.5 Integration by Java Applet Extension

The most simple way to integrate a visualization application with AwareDB is to extend a class called ClientApplet and to implement an interface class called Client.java, as it is shown in the figure 4.6. A Java applet forms the interface to AwareDB and contains certain methods that can be used by the application side while developing an environment. After extending this applet, the designer has to catch the messages coming from AwareDB in their applet. Therefore, an interface is created, which describes the void CLProcess(Package pkg) method, where the results from AwareDB arrive and can be made ready for graphical presentation by the designer. All messages are wrapped as content of a package.

**Example:**

To extend the ClientApplet.java class, the application class has to be declared a subclass of it.

```java
public class Application_Applet_Name extends ClientApplet
    implements Client {...}
```

63
To receive any information and results of queries sent by the AwareDB, a `CLProcess` has to be implemented as a method of the visualization application class.

```java
public void CLProcess(Package pkg) {...}
```

### 4.6 Summary

Taking a three-tier model for the implementation of the forthcoming database, a foundation for the overall connections and distributions is realized. It ensures information exchange among users and simplifies further system extensions including web-based and 3D immersive environments. To operate the forthcoming behavior of the system, this implementation allows actions on the database schema to be made by both application designer and users through the interfaces built at the data modeling components layer. Users’ expression about their preferences on information are allowed and discrepancies between the application designer’s and users’ expectations are shown. Finally, data preparation and parallel access were implemented in the cache component layer and improved the concept of the forthcoming database.

This realization was aimed at achieving the ”forthcoming” behaviors of a system at the technical implementation level in a manner that is strongly related with the data supply. This system, AwareDB, can be further extended so that each of the forthcoming behaviors could be built as extended modules and attached to when possible. In order to do so, the communication and distribution system can be easily reused as they are currently defined.
Chapter 5

Applying Forthcoming Database

In the past chapters, the new role of the database systems in information visualization was emphasized and the importance of taking into consideration the requirements of both the user and application designer with respect to awareness in the environments into account was stressed. To actively facilitate the designing, building and management process of information visualization applications, a framework was proposed that could be used in connection with a database system. This thesis claimed that the concept, the forthcoming database, should allow the database system to become an active supporter of work processes rather than a static container of information. As the result of using the framework of this thesis, application developers should be helped in a forthcoming way with data handling possibilities for their work. Finally, the database can be forthcoming to the user through the application as well.

In this chapter, three case studies in which we were involved throughout the whole process of design, implementation and management are described. The experience through these case studies provided a basis for the evaluation of the forthcoming database’s performance and the results further influenced the development of the concept itself.

5.1 Road Map of Case Studies

The concept of collaboration using a forthcoming database was realized as a prototype. This prototype, called AwareDB (see in the chapter 4), was used in three case studies where it responded to different requirements of the visualization applications. The most important goal of the evaluations of these three studies is mainly to prove how well the designed system performed the intended function on behalf of the forthcoming database concept. Finally, it was also observed what the designers and users gained from it, and these findings were used, in turn, to enhance and further improve the prototype.
## 5.1.1 Three Case Studies

The three case studies the prototype was applied to are as follows:

<table>
<thead>
<tr>
<th>Case</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In a course environment</td>
<td>First, a system of the framework was made in light of the ideas of this thesis and used in a course environment. It served as a communication supporting system for the teachers and students and helped them to be aware of each other’s actions and the progress of their works through the Internet. The main constituents were users and the main goal of this study was <em>analysis</em>. The results of this study were to find out to what extent awareness was a quality that benefited the students and teachers and to carry it forward to enrich the system development. The evaluation process was mainly done after the system was implemented using a list of questionnaires. Thus, it is a <em>summative approach</em> with some amount of <em>quantitative and qualitative questions</em>. The questionnaires, see in Appendix, were accompanied with observations made by students using the system during the course and came up with a list of praise and complaints.</td>
</tr>
<tr>
<td>2. In an Internet-based Virtual Office</td>
<td>The second study came about because the application designer asked to use this framework to solve the problem he was facing building processes in an application. Virtual office was already an existing system and the impact of the integration of the prototype could be observed. The main goal of this study was to see how easily the prototype could be <em>adapted</em> to an existing system and how well their specific problems could be <em>solved</em> by helping the application designer. Certain functionalities of the application served as the study scenario. There were two steps of the integration process. The first part of the integration, which involved binding two systems, was a <em>summative-qualitative</em> evaluation. The second part, in which certain functionalities were added into the prototype, was a combination of the <em>formative</em> and <em>User Cost-Benefit Assessment</em> methods. Finally, some questions were asked to the developer of the application for his subjective rating about the collaborative work result as well.</td>
</tr>
<tr>
<td>3. For an immersive 3D environment</td>
<td>The third case study’s playground was in a new project in which information architects worked for applications to visualize information in an immersive environment. First, the unique needs to this environment were analyzed, since the programming bases differ. Based on the requirements analysis, the prototype was extended in parallel to an application implementation having a scenario. This case produced a very strong <em>formative-qualitative</em> and <em>User cost-benefit Assessment</em> evaluation.</td>
</tr>
</tbody>
</table>
5.1.2 General Paradigm of Existing Evaluations

There are many kinds of evaluations, differing in a wide spectrum that corresponds to the different expectations of evaluation results. Also in this work, the expectation and the new concept of “forthcoming” made it difficult to take over an existing evaluation method one to one. However, a method was shaped for this system using existing paradigms in CSCW and software engineering, which we came across in this research. Next, a brief explanation about the general classification of evaluation is given and then a look at the state-of-the-art with regard to interesting and relevant studies in the fields of human computer interaction and CSCW is taken.

Formative - Summative, Qualitative - Quantitative

The two major categories of evaluation paradigms are formative and summative, which is a distinction determined through the formulation of the research method. An evaluation is formative when it happens during the design process and results in improvements of the system. It focuses on what a system can do and allows for specific improvements “during” design. An evaluation is summative when it happens “after” the system is ready-made and used to judge the results that can be achieved with the designed artifact. The formative evaluation is often used for developing systems of commercial or research purpose. Thus, the evaluation can be used for making a system more usable and appropriate for the intended or actual users. In practice, it is said that the formative evaluation goes seldom beyond usability questions [Nie93]. The focus of this evaluation is on “how” users actually use the system in a given complex context such as an organizational one. Further distinctions include the category functions. Quantitative evaluation is an assessment process that answers the question, “How much did we do?” Thus, the values of the measurements are usually scores and numbers. The qualitative evaluation approach, on the other hand, is an assessment process that answers the question, “How well did we do?” Content, quality, efficiency of strategies and activities and costs in relation to what was achieved, etc. are usually measured [GHD].

The issues that surround the evaluation of Information Technology systems have tended for the most part to be determined by two broad and contrasting styles that use the above evaluation types. These are the summative-quantitative-controlled experimental forms versus the formative-qualitative-opportunistic evaluation [RTB96]. Depending on the researchers’ focus and how the researcher outlines her or his research domain, the expectations through evaluation cover a wide spectrum. Therefore, there are suggestions to apply both approaches ideally with regard to research design and strategies [And96].
Integrated Approach

In designing interfaces for the interaction between users and equipment, an “integrated” approach is emphasized [And96, AvdV93] in which three constituents are concerned: individual-environment interaction, human interaction at the organizational and group level, and media theory. It analyzes the way individuals use the system and the influence and effect of the system on user interaction and interpersonal communication. For proper evaluation, the following steps are proposed that summarize integrated evaluation theory in analysis steps regarding intensity and frequency of system use, system impact, and the characterization of constraints and requirements for an optimal system:

1. System characterization including the quality of the infrastructure
2. Input elements description
3. Intended success criteria description
4. Evaluation and result comparison regarding aspects including the improvement of the communication process, system purpose, etc.

User Cost-Benefit Assessment Method

Making a mixture method out of the traditional evaluations logical-positivist and interpretative, Eason and Olpert [EO96] set their argument as follows:

*The successful application of CSCW systems is much more dependent than other forms of computer systems upon developers being able to predict and plan the organizational implications of their systems*

The logical-positivist approach is based on an effect analysis by the developer when a new system is introduced. The analysis is made with regard to the organizational structure and specific processes going on, and the organization gets modeled in such a way that the impact of technical change can be determined. The interpretative approach is one of the families of user-centered design [GLB99] methods and emphasizes the collaboration with users and/or stakeholders. The philosophy underpinning this approach is that effective systems are created by a partnership between developers and the users and/or stakeholders in the organization which is to operate the new system.

Based on these two approaches, Eason and Olpert suggested a framework with stages:

1. Describe technical system including goals, functionalities and its data structure
2. Analyze preliminary
3. Create a usage scenario
4. Evaluate the scenario in regard with benefits and costs
5. In case the first result is positive, then carry it forward to detailed technical design and to the formulation of the plan.

In this framework, the stakeholders in an organization can predict the implications of a development and assess them.

**Usability and Utility Evaluation**

Usability is the mostly chosen methodology by practitioners in Human-Computer Interaction (HCI) and Computer Supported Collaborative Work (CSCW). The reason for its frequent use is the clarity of variables of this method. Most usability considerations are related to computer software applications and deal with system acceptability issues, see figure 5.1, or whether the system is good enough to satisfy all the needs and requirements of the users and other potential stakeholders, such as the users, clients and managers. Traditionally, five attributes are associated with usability: learn ability, efficiency, memorability, errors, and satisfaction. This applies to all aspects of a system with which a human might interact, including installation and maintenance procedures. Thus, usability is not a single, one-dimensional property of a user interface, but rather it contains multiple components even though researchers using this method usually speak about the interface in a broad sense. Usability requires the interdependency between the design of the tool and its components, including users, tasks, environment and the user support provided, and the breakdown between a human and any component of the human-computer system interaction is seen as a usability failure [SUP96]. Compared to usability, which asks how well users can use the system functionality, utility is a question of whether the functionality of the system can do what is needed [Gut97], and it must be treated independently from usability in order to promote flexibility of the system [Gru92]. These questions together finally lead to the system acceptability question.
Measuring Awareness

There are already techniques used to assess awareness, including the Situational Awareness Global Assessment Technique (SAGAT), Situation Awareness Rating Technique (SART), Advanced Medium-Range Air-to-Air Missile Operation Utility Evaluation Test (AMRAAM OUE) and Subjective Workload Dominance metric (SWORD) [Jen99]. As the titles lead one to assume, they are mostly concentrated in the aircraft environment. For that reason, a proposal of Endsley [End95] for a more general perspective was reviewed. In general, three different methodologies may be implemented: physiological techniques, performance measurement, subjective techniques and questionnaires. Physiological techniques approaches try to determine whether information is registered cognitively by the subject. Thus, it could be helpful to find out whether the elements in the environment are perceived and processed. However, the amount of and the correctness of the information and derived comprehension are not determinable. For performance measurement, specified performance data for evaluation is automatically collected by the pre-made program in most cases. Therefore, it is highly non-intrusive and objective. However, its limitations are that often the performance measure (global measures) gives only the end results and so, this method could overlook other factors, which may ground a poor performance. Examining performance on specific operator subtasks that are of interest (imbedded task measures) is understood to be more meaningful than other measures, but with possible difficulties in determining appropriate measures for certain systems. This method’s disadvantage is that it lacks the interactive nature of Situation Awareness subcomponents whose improved performance may result in a decreased Situation Awareness on other ones. The self-rating method, so-called subjective techniques, allows people to rate their Situation Awareness themselves. The advantage of this method is its low cost and ease of use. A second type of subjective technique is the observer-rating method. It involves independent and knowledgeable observers to rate the quality of a subject’s Situation Awareness. In both cases, the ability of the operators or observers can be put in question. Questionnaires allow one to gain detailed information about Situation Awareness. It can be collected on an element-by-element basis that can be evaluated against reality. This type is a more direct measure and offers a complete set of information to the operator. A detailed questionnaire can be administered after the completion of each simulated trial, on-line during the trial, or during breaks put artificially into the trial.

5.1.3 The Evaluation Directives

The case studies’ goal is to support systems aimed at data visualization with the additional concepts, including enhancing user awareness. For the system evaluation through three case studies, we take advantage of the actual tendency of the evaluation to integrate different expertise and perspectives, and in so doing, tailor the
directives and set questions to make them as tangible from the paradigms as much as possible.

Having in mind that design, use, and evaluation are intertwined and interpenetrating practices and not distinct steps [Ban93], the directions are put into concrete terms that will accompany the observation steps in the next sections 5.2, 5.3 and 5.4.

- A description of the visualization application environment purely from the visualization aspect could give us an exact picture of how this framework could be used for visualization applications. According to the integrated approach and user cost-benefit assessment method, the following points could be included in the description:
  - User and data description
  - Objectives of the application from different perspectives
  - Application designers intension
  - Preliminary analysis about the quality of the application infrastructure and intentions

- Scenarios of the application that show it in use are helpful to look at the usability issue from the aspect of the visualization. It further gives us the idea of interactions that reaches to the original database content and also gives insight into how a self-rating could be built in the application to gain information that is of interest. It especially shows us how the data structure is visualized on the application side and enables us to compare the object difference.

- How well the resulting system integrated with the framework meets the objectives of the application it is focused on. As the cases differ from concept to implementation, questionnaires could be also used for the user and application designers to evaluate the utility of the framework. The following points could be included in the questionnaires:
  - The application system as whole
  - Users’ awareness
  - Developers’ awareness and her/his work
  - Application developers satisfaction using the framework
  - Collaboration between the user and developer, developer and the system

According to the paradigms above, the overall approach in the evaluation could be cited as a *qualitative-formative-utility evaluation* [RTB96]. As the focus was put on how to prove whether the designed framework performed the intended function for
specific visualization concepts or not and what the designers and users gained by it, utility evaluation and iterative prototyping also came into play.

In the next three chapters, the application of the framework prototype, AwareDB, will be investigated in a “real application world”, which encompasses different contexts and construction activities.
5.2 Awareness Support for web-based classroom environments

This first application was an exploratory study looking at the effects of different information provided for the course participants' works. The course platform was developed, an extensive website, which consisted of a database system, the framework prototype AwareDB, and a web-based course environment providing several interfaces displaying teams and works carried out by the course participants. The study comprised the analysis of the process of developing a lecture environment, the observation of its use by the course participants including students and teachers, and its evaluation in order to identify users' satisfaction and acceptance. This was also an important experience regarding the definition of designers' needs in providing awareness information in a lecture environment.

The main findings of this study show that participants found the environment to be useful, as it provided awareness information regarding the work processes carried out during the course and also developed by the other teams. Some design requirements were also identified: transparency by providing compressed information and events, connection flexibility between the back-end system and the implementation layer for the visualization, and a back-end system that facilitates the availability of a data awareness application on the Internet. The framework in this application provided the possibility to convey some event information that is a part of awareness information. On behalf of this application, a functionality to give the application designer model awareness information and prove the stability of the communication channel handler of AwareDB was finally designed.

This section starts with a description of the goal with respect to the lecturer’s philosophy. Then, the course interfaces are introduced, which were implemented using the framework as a base. The resulting findings are discussed with regard to awareness information and AwareDB.

5.2.1 Goals of the Study and Data Collection

What follows is a summary of the lecturer’s general understanding about the course, in which the framework forthcoming database was applied as the basis of an Internet-based classroom environment.

In computer courses, most of the work usually happens “behind the screen”. The monitor becomes the window that reveals views into the working environment. This fact emphasizes the importance of choosing an appropriate design for the interface, which becomes as important a part of the learning environment as the lecture hall or the design studio. Therefore, the visual qualities of the interface have to match the intended qualities of the content of the course and imply the qualitative expectations for the contributions by
the students. All the work done by the students will appear visually on the interface. The interface must therefore be able to function as a very strong framework. (From [Eng98] → Visual Qualities, Graphical Interface and Navigation)

According to these considerations of the lecture designer, the overall aims of the study were set as follows:

- Identify the kind of data which is important to help users’ consciousness of this course environment, or awareness information.
- Provide the awareness information and analyze its acceptance by the students.
- Gain experience with the implementation process of such an awareness system and find the best way to facilitate the designers’ work for the development.
- Prove AwareDB as an effective application framework for data display. Furthermore, consider the requirements to support collaborations supposed to be placed in the course environment.

AwareDB had some of these basic functionalities ready at the time of the course start. The communication channel handler to the database and among users could be used at any moment. The awareness and application data handler were figured out conceptually, but their realization as functionalities had not yet been further studied in-depth. Corresponding to the application designer’s work, this course environment was developed, and it was expected that the result could bring the concept implementation to a higher level.

This case study considered the idea of setting the quality of the content equal to the visual quality. At the same time, the importance of a strong back-end system or platform to support such lecture environments was generally recognized from experience gathered over several years. Thus, a system was prepared that could be flexibly modified according to the evolving course concept and students’ requirements.

This study used observations, comments, questionnaires and small discussion techniques to gather the course participants’ feedback. The users had to follow a set of steps during the course, which included having to use the system to present their own work, to access others’ work and to communicate with teachers and other students. However, there was no control of how they would use or what they would use from the system, with the exception of the upload page, which is not related to the evaluation. To submit their work into the case study Internet site, they had to use an html form, i.e. the upload page.

Specifically, information requirements, usability, and design issues were looked at. Evaluation of the specific information requirements involved three questions: how
often the participants used the information, which information was required to pro-
vide awareness in this environment, and how they searched for specific data within
the environment. The investigation of usability looked at how displaying awareness
information affects people’s work. The queries were whether the information pro-
vided inspired their work and helped speed it up, if it was easy to find the desired
information, and if the given information led them to the right source of data. Re-
arding design issues, the following questions were considered: if they would take
this interface as an example or base of their system in case they had to develop a sim-
ilar environment and if the designer’s intention matched users’ expectations. Taking
into consideration that data and its representation are intrinsically connected, some-
times the evaluation covers issues related to the interface display. The interface was
designed according to the specification provided by the teacher responsible for the
lecture, and in order to facilitate the data manipulation, a course database was
structured to take into account the interface and information displayed.

There were several styles of evaluation depending on the year. In one semester,
students were asked to freely submit their comments about the course environment
using a form on the Internet. The first evaluation was made by seven students of the
course using an html form. The second one, which was mostly considered for this
documentation, was done by 21 students of more recent semesters. Anonymity was
maintained to stimulate free expression. For this evaluation, the form in Appendix
A was used. The experiences as teacher or supporter of the course, using the sites
for supervision and to provide comments and feedbacks, made it possible to closely
follow the development of the students’ work process and build good insight into the
use of the program. Finally, informal discussions during the course and additional
comments after the evaluation that also provided useful complementary feedback
were registered.

In addition, the course environment allowed us to identify areas of interest during
the analysis. The following sections describe details of the methodology, including
the participants and the setup of the study together with the evaluation result.

5.2.2 Course Environment, Participants, Process

Internet technology has also influenced classroom culture and gained much inter-
est in the area of education [HWV96, STB+03]. Some projects are designed to
even loan laptops to the students for the improvement of the course [Ryd00]. The
development of teaching material for web publication has become increasingly im-
portant, while conversations among students and teachers happen more and more
over the Internet. In addition to its usefulness regarding easy access to content and
communication, an Internet version of a lecture space helps to build an extended
form of the course community. This case study course was tailored for architecture
students and aimed to introduce object-oriented programming paradigms focusing
on developing algorithms that formulate spatial and temporal relationships. There are many trial courses, which have already proposed video streams as course media [CCCH99, LRS03], considered the visualization as lecture technique [Whi01], or provided a web-based browsing system [McG01]. This case study, however, is a course in which the technical materials are tailored for the course specifically and the students are very much involved in visually representing their work concept and the result of their programming.

This study periphery was a course held by Prof. Maia Engeli during the winter semesters at the ETH Zürich, Switzerland [pro] and at the University Kassel, Germany, during the years 1999 and 2001. The number of students attending the course varied each year between 20 to 30 students and they were organized in teams of 2 or 3. The evaluations were freely submitted, thus only 41 opinions were used for the analysis at the end. The functionalities offered differed slightly each year. For example, the functionality to comment on each other’s work was only offered in 1999. The students used the interface to register themselves as a team, upload their work to the system (i.e. to save their work in the persistent database, which would then be shown on the interface), to see others’ work, to comment on others’ work, etc.

The modeling and animation software MAYA was chosen for this course due to the concepts and primitives that it supports, allowing the free exploration of spatial and temporal relations. It has a scripting language named MAYA Embedded Language (MEL), which is sophisticated enough to be used as an introduction to programming. It enables the simple creation of objects such as spheres, cubes, and cylinders, including composed objects, scenes, and also attaching scripts to objects. Behaviors can be easily attached to objects and determine their relationship with other objects and time. The complex concepts of architecture students could be implemented using MEL and the immediacy between writing code and its effect on the scene supported an understanding of programming.

However, this tool MAYA had no connection to the Internet environment to present their work results. In order to provide the work results of students in an Internet environment, a web-based lecture site using AwareDB was implemented and some further system functionalities such as event information distribution were included as HTML links or Java Applets. Thus, the course environment and students’ design work environment were offered parallel to each other, not integrated. The course materials were also on the course website and the teachers used HTML pages during the course. The MAYA environment and awareness view sites could be opened in different windows and kept next to each other.

The course concept was based on “extreme programming”, which induced the lecturer to pair programming with concept definition and focused work. The students worked in a team consisting of at least two people. For each lecture, there were ex-
ercises to do as homework. For example, a student had to choose a scene of movies to illustrate their idea about “notation in time and space”. Starting with an image, they developed the work according to their concept and results along the week.

To concentrate on the scope, they always had to describe their concept first, which helped them to know what was already done and what remained to be done. This process also helped others to understand their work. In terms of communication, they were in the same school and could contact each other, although most of them worked in different places and at different times. Thus, the communication could be via email, using the address link provided by the system.

The participants were from three different categories:

- Students between the first and thirteenth semester
- Postgraduate students who had finished their regular course of study
- Members of the Computer Aided Architectural Design (CAAD) department working as developers and teachers in other lectures.

Around 50% of the students declared themselves experienced in Internet use.

In the first lecture, there was a demonstration to show students how to submit their work, where the results would show up, how to use the small functionalities included, etc. However, much explicit information was not given because the interface should be self-explanatory regarding how it could be used, showing if the student had uploaded their work, etc. Support to overcome technical troubles was also provided.

5.2.3 Set-up for Information Display

All course related information was available through the Internet. The version using Java could only be used supported by the Netscape browser because it used JavaScript. Here, students could register various pieces of information arranged in different windows. “Review” sites containing the following windows for the evaluation were built: W1 for overview; W2 for team works; W3 for each specific work; W4 for event happening; W5 for database news (see figures 5.2, 5.3, 5.4, 5.5 and 5.6). Next, each interface will be described more in detail.

W1: “teams and all work” overview window (figure 5.2)
It showed an entire overview of the teams in the course with their images and names and of all work submitted in one frame. For each task, they could submit several snapshots of their work results and one of these images was shown in this frame to represent each work. This frame was dynamically created depending on the submission date. In the first version of the application, some possibilities were included to see how many comments on their work were given and also how many times the work had been downloaded by others. These were not only open to participants but
Figure 5.2: W1: Overview of the teams and all works submitted. On the left column, three teams are shown and on the right side of each group, works done by each group are shown.

also to people outside of the course community.

W2: “all work of a team” window (figure 5.3)
Works developed by a specific team were shown here together with their concept. It showed how a concept was developed in time and allowed the viewer to follow the idea of a specific team. This was only offered in the second and third versions of the system. Participants could see all of the images through image animation, programmed in JavaScript by Steffen Lemmerzahl.

W3: “a specific work” window (figure 5.4)
Each work of a group developed for a specific task was shown in detail. This showed more explicit data about a work, including code, movie, comments, concept, and all images. The movies were delivered through webcasting streaming video and audio [Pul00], which shows the dynamic result of a program. In the first version, more detailed information was also listed: the content of the comments and the list of users who downloaded the code.
Figure 5.3: W2: an interface showing all works of a specific team. This interface shows works submitted by Johannes and Russel over the 5 lecture weeks.
Figure 5.4: W3: shows detail information about a submitted work including concept description and resulting images.

<table>
<thead>
<tr>
<th>Upload</th>
<th>Comment</th>
</tr>
</thead>
</table>
| creation by 0101  
for exercise 5 with concept numbers floating around in space... | (group: 0101) get comment of Guest for work 3 comment will be.. |
| creation by afm  
for exercise 3 with concept it grows around and rewwed | (group: 0101) get comment of Guest for work 4 comment will be.. |
| creation by bige  
for exercise 4 with concept Main Procedure: WeldGMat, which is creating area... | (group: 0101) get comment of Guest for work 3 comment will be.. |
| creation by BP  
for exercise 3 with concept animation | (group: 0101) get comment of Guest for work 4 comment will be.. |

Figure 5.5: W4: all recent work submissions are shown in the event window.
**W4: “event” window** (figure 5.5)
A short description of the work and the comments submitted during the last two days were listed as events. To distribute this information more effectively, this frame was shown in the first page whenever the student opened the course environment.

**W5: “database news” window** (figure 5.6)
All information presented in W1 to W4 was the result of a user’s retrieval action by choosing a work or submitting one. This window, in contrast to the other, automatically and immediately distributes information from the database system. This is implemented as a Java application and can be kept open all the time. The user could leave it open all the time, even when the user did not have the course site open in her browser.

![Figure 5.6: W5: a Java application window distributing new events in the database.](image)

Inside of these “Review” pages, functionalities were included for commenting on each other’s work, downloading code, and submitting specific code they found helpful in developing their work.

The windows W1, W2 and W3 were used by the teachers to present and collaboratively discuss students’ works during the lectures. There was also a positive motivation to use the environment frequently. The tables in the database corresponded to the functionalities required in these windows (see figure 5.7). Oracle or MySQL were used to keep and maintain this course data. To keep the evaluation concentrated on data content rather than data display, a very linear presentation of components was kept in the user interface: teams - team - work (see figures 5.2, 5.3 and 5.4).
Web-based systems can provide support for learning concepts, understanding techniques and learning programming and problem-solving skills [NBJP+97]. As such, this programming course environment also provides support for the course in educational and pedagogical aspects including the course concept and problem solving skills allowing the users to be actively engaged in publishing their work with the consideration of many different reviewers.

5.2.4 General Observation and User Evaluation

This study investigated the users’ needs for awareness information in an Internet-based lecture environment and the development of such a system. Considering these issues, the study results are organized as shown below. First, the general observations are presented and then the focus will be on the students’ evaluation of the windows provided in the course.

The course application environment provided not only feedback confirming the work submission, but also immediate feedback on how the work results are shown in the context of the entire course environment. The majority of the participants felt comfortable with the provided interfaces and found them useful in order to improve their work process. Participants were clearly aware of who was attending the course and who was doing what. Even though the environment did not force people to exchange information between teams, participants used information from all windows to understand each other’s goals. Furthermore, they used the information to
compare each others' work, which led them to constructive competition similar to the course stimulating the interactivity among students using peer-review [SM03] It was observed that people downloaded codes from other teams to learn and to copy certain algorithms for their use. People still preferred to ask somebody personally for gathering more complex knowledge, for example to solve problems they were facing at the moment. In this context, the interface was frequently used as a guidebook to find out who could have the data they were searching for.

Teachers did not need to be involved in the submission process to collect the works of the students. However, the teachers used the interface to find out which teams were having problems and to check whether participants understood the lecture content. Furthermore, they could give advice to participants within the environment, even before the students realized that they were doing something wrong. Some windows such as W4 and W5 were often used by the teachers, assistants and the system administrator to monitor student action, analyze problems, and be aware of student work activity over the night, for example.

Students were interested in receiving peer evaluations of their work. However, they did not feel comfortable giving their own comments on each other's work. Some groups used the comment functionality to cheer themselves up by submitting positive comments about their own work. In the end, the evaluation results showed that information and opinion exchange among participants was simplified by the use of the provided awareness information, which, in turn, was helpful for the development of the students' work.

Technical issues required the participants' patience and effort. The most frequent problem at the beginning was related to format: An uploaded image in a format that is not JPEG or GIF cannot be browsed by a web browser, and as a result, is shown as a broken image. This type of mistake happened over the course of the first two weeks after having started the lecture. Some students complained that the site was very slow when opened from their home using a modem connection. The reason for the slowness was the number of images and also their concentration on the overview page, window W1. Some students argued that W5 did not work in certain browsers and did not stay in the foreground position and, therefore, they forgot about it after a while. Even though the technical limitations of the Java language has been explained, users tended to try to interpret or use it how they wanted to. As in CoMMIT [ECRG⁺00], it was observed that the person who used it more frequently did not report this kind of negative experience.

Some students manifested the need to access and control certain data in the database. They did not know what was behind it necessarily. However, it was convincing that the user be given some more freedom to be able to control data, especially when it could safely reduce administrators' work.
In order to evaluate the “Review” pages, participants were asked to express their opinions. Their feedback was organized in topics as shown below, followed by graphic charts and comments.

**Course Description:** The course participants felt a strong need to share information and have collaborative discussions during the course, as shown in figure 5.8. However the need for work distribution within the teams was not cited as being so relevant, as the course was designed only for two or three person groups.

![Figure 5.8: Course participants described their use of the course environment.](image)

**Design:** The course interface was evaluated as a good example for a course. It was said that if the users needed to build a similar environment, they would take the system as an example and as a start framework.

People evaluated transparency as the most important feature in such a system and rated privacy as being less important. Other information on activity, work results, progress, and work stages were said to be important features to be provided.

**Information Search:** To search for information about other teams’ work progress, participants mostly used window W1, followed by W3 and W2. A lesser-used approach was to ask somebody directly. In case they had certain programming problems to realize their concept and had to find necessary information, 71% of the students chose face-to-face conversation, an absolute majority (see figure 5.9). Each work could be opened on the window type W3 and detailed information about a work, including code, could be found there. This page, however, was only used by 66% of the participants to get codes that could be helpful for their concept development and brainstorming.

**User Satisfaction:** The mostly used window was W1, which showed a general view of all information including teams and works together in one page. “Teams’ work” page W2 and “specific work” page W3 were used moderately by 61% and 67% of the participants.

More than 90% of the participants agreed that by using windows, people could be
very well aware of their own and their colleagues’ work development processes. 73% of the participants said that the information in the “Review” pages inspired their work and also the access to images, program codes, movies etc. helped them to solve problems and speed up their work.

Synchronous and Asynchronous Data Delivery: Only three people said that the system’s lack of synchronous behavior disturbed their work. Event information was declared as the most necessary information to be immediately delivered through synchronous communication.

These were answers to specifically tailored queries (see the query form in the Appendix, including information search and design aspects). More details about advantages and disadvantages, and other free comments are presented in the following two subsections.

5.2.5 Analysis and Design Decisions

The analysis is summarized by the following seven points:

1. User Experience and Information Use
The students’ experience in Internet use was quite varied. There were students who learned to make an html-page, such as a homepage, for the first time during this course. This means that the Internet experience of some participants was somehow limited to really understanding how the website works. Some of them, however, were already very experienced in programming using scripting languages and relational databases, including MySQL. However this wide spectrum of participants did not seem to influence the information access style of the users. According to
the users, the frequent use of window W1 was due to several factors: fast overview, suitable interface for the course, nice and unitary arrangement of data presentation and amount of structured information.

2. Interface for Work Process
The chosen simple visual interface was often praised as being fast, simple, good looking and clear. It was said to be a positive fact that all works were presented in the same format. But on the other hand, it disturbed the flexibility of the functionality including the possibility to submit more data than predetermined.

All three windows seem to have been helpful for the users' work development. It was especially noted that they helped participants during the work process: brainstorming → sketching → schema development → work submission. The mostly praised features of these windows were transparency and ease of use. From a work-related perspective, they found information about activity, including submission, work results and progress to be important features. Participants cited transparency and discussion support as the most important features of Internet-based lecture environments similar to this course.

The work presentation format did not allow a personalized presentation of each work. It allowed only the same pattern to be used for all the groups. Therefore, participants seemed to feel more pressure to think about how to present their work in a way that would be easily understood by others. Therefore, they focused more on concept description and also on the careful choice of images.

Despite emphasizing the usefulness of the environment, some users criticized certain aspects. The representation of image and navigation were not distinguishable from a distance. For example, an image that looked like a program result in window W1 turned out to be a snapshot (e.g. simple image) of an animation in window W3. Another critique was that a specific window was not identifiable after it was left open for a while, since the windows W1 to W4 had similar structure and colors.

3. Monitoring and Exploring
Windows W1, W3 and W3 were used more often by students and W4, by teachers. The teamwork page W2 was included only during the last two semesters. It was used much less than W1, but it was helpful for a teacher to see how a team developed their concept and how it was worked out over a long time span. It was also good for each team to show their work in a line and to have a clear idea about their own work progress. This window was the only one, which was used more by users with less experience with the Internet than by the experienced ones. It could be that the inexperienced user tended to explore the environment more than the experienced ones, who felt more confident about what they saw and knew and were aware of other more sophisticated possibilities.

An option for users to discuss with each other using synchronous communication was not provided. However, certain actions of other users that were causing database change were published on the database window W5. Contrary to expectations, the
students used it very rarely, but it was helpful for teachers to monitor student actions. Window W5 could also be left open overnight and the next morning one could note who had submitted work in the meantime. In the rare cases when no work was submitted, the students could be immediately contacted to find out if there was a problem. The users’ response shows that synchronous information would have been useful for communication throughout the course.

4. Privacy and control on own data
The public workplace was taken seriously and the users wished to be in control of it. They felt they should be able to manipulate and change their work whenever they wanted to do so. Surprisingly, privacy was not understood as an important feature. The reason could be the fact that the participants could have their own account in this course where they kept their work in private files. Therefore, submission seemed to happen only after thoughtful consideration. Sometimes, restrictions related to the state, including license conditions for certain necessary tools and availability of system support, led to the lack of personal work areas. In these cases, students asked us to ensure more privacy.

5. Awareness information of others
More than 80% of the participants wanted to be aware of other teams’ goals, actions and interests. The presence of other teams was only considered important by 50% of the participants. The “Review” sites somehow forced the participants to present their intentions, which were described as concepts (e.g. goal). Participants found that window W1 was most useful for allowing them to be aware of the others’ and their own work. It was thus convincing that overview information has substantial value in individual work. However the details were to be available as well. One student noted, “Whenever I wanted to have more detailed information, I could get it easily....”

Regarding transparency, there were some efforts to offer and combine information as much as possible by using html links. For example, the lecture content could have been put in the “Review” site too. A path was considered, which could lead people to a more meaningful line of information. This brought us into a complicated retrieval effort at the implementation level in order to match the visualization structure. In contrast to the intention of this case study, some users felt that there were too many steps to access the data they wanted.

User awareness developed also a kind of competitive spirit among the students. Some students felt disturbed by the empty columns, which represented not submitted work. They could see which teams were slower and some teams felt motivated by seeing fewer filled columns on their space when compared to the others.

6. Communication with others
Work page W3 was used to look at a specific work with its codes, comments, etc. in-depth. The comments from others definitively motivated people. A comment
sometimes became more than one direction communication and developed into a
discussion between teams. Program codes were unfortunately uploaded almost on
the same day, because there were submission deadlines from the course. Therefore,
students did not have much time left to wait and see others’ work and use them as
examples. Participants used these examples later to see how certain things worked.
In case of a one-day workshop that was part of the course, they stayed the whole
day in the same room and thus, had more face-to-face conversation rather than
searching code via this window.
Participants found it easy to contact other participants. In this environment there
were no specifically designed tools for that aim, such as chatting tools or Instant
Messenger. Email links to each team, team member, and teachers were offered and
this seemed to have been enough. There were reliable connections to the database
and inside the system. However, video or audio based communication were not
offered and technical problems with lacking reliability and quality of synchronous
Internet-based communication [EZ98] were not faced.
Participants were especially interested in getting some feedback from other partici-
pants, not only from teachers, to get an objective image of their work. The comment
information shown in windows W1 and W3 was used for this, with the exception of
one semester, in which this functionality was not included in the interface. During
this specific semester, students complained about the lack of this functionality.

7. Request for immediacy
One student requested to use communication tools such as Instant Messenger, which
gives information about online presence. He said that he sometimes had urgent
questions and wanted to know whether the assistant was around to be immediately
consulted. He said that when writing an email there was no information about the
recipient presence at that specific time. With the exception of the discussion data,
all information the users considered found as important to be synchronously deliv-
ered was related to work: schedule, homework, codes, actuality etc..

The study showed that the “Review” windows were fairly useful in a course en-
vironment to promote course participants’ awareness and generate new input for
their work. As mentioned in the above summary, this study has exposed some of
the underlying issues including awareness information handling and the designer’s
work.
According to points 1 to 2, the interfaces visualizing information should include
many different aspects that might greatly vary depending on the application and
concept. However, the most important fact seems to be transparency, which guar-
antees fast overview from both application designer (teacher and administrator) and
the user. This transparency seems to be the base that should be provided as much as
possible to the application side, as it stretches over from point 1 to 7. Furthermore,
transparency is the key factor that is able to move this kind of course environment
up to a collaboration environment.
This thesis also concludes that user awareness is not only provided through actively showing specific information to users. It is more effectively provided through the visualization of overall information of the application data. In addition, with respect to point 5, user awareness should be confirmable by browsing a simple display of information too. It does not necessarily need a specifically tailored form of window, such as the interface W5.
Similar to this application, other interfaces might have a very different concept of the data visualization and could also be used differently by the users. Thus, the discrepancies between the application designer’s and users’ intentions should be also worked out respectively inside of the system. Or the application designer could get the information by monitoring use and trying to be forthcoming to users’ desires by tuning or changing the system functionality.
As a result of this study, the functionality of the Chapter 4.2 was implemented, including visible database and interest setting of users and application designers.

5.2.6 Summary
In this chapter, a study on awareness support in a classroom was presented in which participants used the system to accompany the lecture, engage in mixed focus collaboration and influence each other’s work. Consequently, this study provided valuable experience in the definition of awareness information needs, the corresponding system development, and user perception. It raised issues that were taken into consideration during further developments and experiments. This study proved the usefulness of the communication layer of AwareDB and showed that it could be used successfully over the years. Furthermore, the results led to the design decision to build new functionalities in the object handler layer of AwareDB.
5.3 Virtual Office - Laboratory Design Experiment

In this chapter, the application of the prototype, AwareDB, to an Internet-based office environment called Virtual Office is described. The Virtual Office environment was taken as the second case study because of some specific needs pointed out by its developer. This included a communication network architecture connecting all users and the database that could be provided by AwareDB. As a result of the lessons learned from the last study described in Chapter 5.2, a much more mature version of the prototype could be provided for this case study.

With the Virtual Office environment, it was proved that this system can offer a kind of application platform in developing an Internet-based system. Furthermore, much insight could be gained into the processes involved with how a developer could prepare, offer and manage awareness data in a group application. One of the main components developed through this study is the extended facility for handling awareness data in the prototype AwareDB. The findings show that the prototype assists work using the following process:

- Building overall communication network in a group application
- Offering users opportunities to describe their own interest
- Observing users’ interests and their influence on the application designer’s decisions
- Maintaining the application system so that it reacts to up-to-date user information with the help of the database system

Users’ comments and developers’ evaluations of the Virtual Office proved the positive influence of the framework system on an Internet-based groupware application.

5.3.1 Goal of the Experiment

Virtual Office is a typical Internet-based information environment briefing people about their presence and the ability to communicate with each other. The World-WideWeb is understood with regard to the awareness as an implementation platform for an awareness system and as an activity space that people should be made more aware of [Lie00]. Virtual Office belongs to the category that takes the WWW as such an activity space with presence and schedule information. However, taking the WWW as an implementation platform was not yet successfully managed when it was implemented and already in use. During use, the developer recognized a problem with its communication mechanism. The communication to the application database was well designed in the entire application, but the automatic distribution channel back to the user side and also a channel between users were missing. This problem was simply described by the developer’s comment,
The developer was also working on an additional functionality of scheduling holidays inside the existing Virtual Office application and wanted to redesign it so that it takes users’ interests into account. For these demands, the developer decided to use the prototype AwareDB. The first goal was to overcome the communication problem, which was also hindering further development of a well-connected application environment. The second goal was to help users in declaring their personal interests and administrators in flexibly responding to their change of interests. According to these reasons, the directives through this case study could be summarized with the following points:

- Adaptation and facilitation: This case study was an application on an existing system. Therefore, it was essential to know how easy it is to adapt AwareDB to an existing application. It should be as convenient as possible. Most desirable is that the prototype and the application fit together technically so that it can be attached as a parallel module for required needs.

- Database insight and evolving interest: Parallel to the application, an interface to the database content designed as a module is offered, which is easy to integrate as a link. This is provided by the AwareDB functionalities implemented modular-wise. The control by people was allowed only in a limited way and the control on data with regard to the developer’s work. This data was represented as part of the content of their application. However, the intention was to offer the same data in a different, less manipulated level and see how people interact with it to express their interest. It was also expected to observe the process of collaboration between developer and users as their interest evolved.

- Influence on development and use: Providing AwareDB for this case study, many positive influences were expected, including reduced work and re-consideration about database needs and the way how the database could be better used. Some of these points are related to the development of the software and some others to the experience of users. It was of interest to observe how AwareDB can influence the application designer’s work and the application itself.

We would like to construct this chapter by first presenting a description of the environment, the methods involved, and finally the evaluation results.

5.3.2 Virtual Office - Visualization application informing where we are

First, a short general overview about the application considered in the next section will be given. Virtual Office is an Internet-based application developed by
Benjamin Stäger for a group in a dynamic work field with numerous projects and goals. Its aim was mainly to provide a common space for project members who are at different locations or traveling. It shows the members who are currently online and helps users to contact on- and offline members through different means of communication [SE01a]. The Virtual Office shows presence information, allows for the influencing of members’ work, and promotes socially coordinated conversation among users similar to the WebWho system [La00, aL02]. In recent years, this kind of virtual office environment supporting short message exchange and presence awareness has gained much interest from researchers [DRS00]. The Virtual Office differs from the others, because it makes an effort to keep the balance between open and closed environmental characteristics in one web-based system environment. It offers visualized connections and information about the presence of people by allowing them to present graphical and personal information. Further, it invites people to join the group and still guarantees secure handling and exchange of project information among members.

In this thesis, a practical example is considered, which is a part of the entire virtual office system. It represents an internet version of the group “CAAD and Architecture” in which more than 30 members are registered, as shown in figure 5.10. It shows the offices of a physical building and the presence of people in specific rooms. Using different colors outlining each member’s picture, it shows the probability of being able to reach a person in a specific room. People who worked in the group or people from outside could also come into the Virtual Office and get in touch with this group for any reason, with or without a common project.

Each user has its own corner where they can put information about themselves including their telephone number and weekly schedule that will be accessible by others. As a communication instrument, it provides tools to send a message to everybody sitting in the same room or to a specific person. The presence information is recognized often as an important component of collaboration [GHJL00]. The concept of the Virtual Office can be shortly described as, “Let people be aware of each others presence through the Internet environment” to help them in their collaboration. As a typical Internet-based middle size groupware, Virtual Office was designed as simply as possible. It was also designed to be flexible enough in relation to back-end systems such as databases or applications that use it such as a browser. The adaptability of the Virtual Office was thus an important criteria set by the author.

The implementation of Virtual Office was done using PHP to connect to its database in MySQL, and for the web presentation, HTML and JavaScript were used. There is ongoing work on a browser independent version of Virtual Office.
5.3.3 Communication Problem and Needs of Database Action

The Virtual Office described above became the second case study field with the developer’s interest in improving his system. At the beginning, the prototype gained the developer’s interest because of the efficient communication mechanism, which was lacking in Virtual Office and had potential for adding further advantages for better information handling in general. Later on, the interest of the developer in AwareDB increased because of the possibility to deal with users’ preferences in his application.

We would first like to describe the specific needs of Virtual Office more in detail followed by a description about how AwareDB was used as a solution for those problems. Thus, this chapter contains the integration process of the prototype in an existing groupware application as a communication mechanism and functionalities for users’ interests on specific application data.

As the developers’ work was accompanied for the most of the time, special attention was given to minimize the designer’s effort, and in exchange, his comments could be used to improve the prototype.
How to inform about changes

The website of the Virtual Office was generated out of the data read out from the database placed on a server. The connection to that application database was mainly done using a PHP-engine on the server side. Thus, the lack of a substantial bi-directional communication channel soon appeared as one of the main problems of Virtual Office. It caused inconveniences in broadcasting and providing information to all logged-in users. For example, if a user entered the Virtual Office environment using a given login process, the information about this user’s entrance was saved in the Virtual Office database through a PHP database query. In response, the newcomer got the list of the people working in the Virtual Office at that very moment. On the contrary, users already in the Virtual Office could not know about newcomers, because the server side did not know how to connect back to these users to inform them. To solve this problem, PHP queries were sent by the Virtual Office environment at regular intervals to retrieve users’ presence data. This increased transaction rate, but Virtual Office still provided wrong information about users’ accessibility depending on how large the interval was. In this case, the entrance time of a user shown in the Virtual Office was the time when the SQL-database query was sent, not the time when the user really entered the Virtual Office. For example, the entrance time of the user Pereira in the room is visualized on the interface as shown in the figure 5.11. But this might have already happened one hour before depending

Figure 5.11: Shows the entrance time of a user.
on the time when the query was done. Consequently, the probability of reaching a person as shown on the interface failed due to wrong information about the user’s entrance in a certain room. Thus, the fundamental problem could be reformulated as “How can we let the system know about the current changes”?

How to let users form their personal interests

The members in Virtual Office made an effort to enter in the environment wherever they were. The group was not seeing each other everyday in a same office and many members felt more freedom working at home or somewhere else. This may be the reason why people were motivated to be in the Virtual Office without any pressure from the outside. However, the Virtual Office was not the main field of work, even though work-related communications could go on through the environment. As a result, the interface environment was often put behind the other interfaces. The interest to be actively informed in such a situation was often expressed by the users as well. However, the users did not wish that the environment popped up in front whenever new information arrived. The main users of Virtual Office were very much experienced in Internet environments and also in developing group application systems using databases. Accordingly, they were quite sensitive about their preferences. Virtual Office had functionalities for setting personal preferences in relation to a graphical outlook. Changing background colors and menu bars, the visual control aspect could be satisfied quite well. In contrast, the support for stating their interest, which influences the data distribution mechanism, was poor. Virtual Office itself proposed a way to represent application data. But it did not then have a function that allowed a normal user to designate their interest on application data. Thus, we had to consider, “How we can let people form their interest on specific data events?”

The needs of Virtual Office were then defined with two points: first, how to inform about changes and second, how to let users form their personal interests. Our goal was to refine the system in a way that it would react at the right time to the right person. Next, the scenario was set where the prototype was applied and provided a solution for these issues.

5.3.4 Technical Adaptation and Designer’s Work Process

To apply the prototype into Virtual Office, the two following steps were processed:

1. Making the Communication Structure: the communication system was integrated to build a network for information streams in the Virtual Office

2. Adding functionalities for users’ interest: functionalities related to user interests and the mechanism how it affects the distribution process are added to a sub-application of the Virtual Office, “Holiday Planner.”
For these two steps, AwareDB first had to be set up. It was provided as a form of Java package: Mediator.class and aware.jar. The former one is a Java application handling database connections. The latter one is a collection of Java packages representing different modules of AwareDB. The location where these files are placed also has to be declared in the "-classpath", so that the packages could be used for the development of each application.

```
javac -classpath /usr/java/lib/classes.zip:aware.jar:. VirtualOffice.java
```

It describes where the packages are to be found and how the application is called. In this example, the application name is VirtualOffice, and the developer used the package aware.jar for the compilation of their application.

### Making Communication Structure

The integration of the system into Virtual Office was not always trouble-free. The biggest difficulty was caused by the discrepancy between the tools used for implementing both systems. The communication to the database in Virtual Office was done directly by the dynamically loaded PHP processes. The HTML results pages were created right away through the PHP processes and retrieved by the client side immediately. Thus, to have a link to all these communication flows between client and PHP server side without changing their PHP code, a module was added in the prototype that can apprise on going queries in the Virtual Office. The module serves as an external service of the Virtual Office and connects the prototype to it.

For the communication channel that has to be persistent for one client session, Virtual Office added a Java class that is an extended class of the prototype and was started in parallel behind the application. For the implementation of this solution, the Java Servlet [Dix97] gained our attention at the beginning. It is the result of server side Java development, which should allow many flexible possibilities of extending and customizing any Java-enabled server function. However, it was decided to stay with a server side Java application and Applet to avoid additional installation effort for an add-on server engine supporting Java Servlets. Also the weak/distributed development stage of functionalities for the communication between PHP and Java Servlet at the moment of this implementation was an additional disadvantage. As a result, the AwareDB system architecture was kept as it was and extended with a PHP Module, so that the Virtual Office and the prototype could be connected with each other.

Developer’s works for this step were as follows:

- Wrap the application in Java Applet using AwareDB packages.
  For this, a template showing how the Applet could look like was offered. Using this template, the name of the class could be changed accordingly to the
developer’s application and added inside of their application. After this was done, some essential methods could be used including connecting a specific user verifying the database content and reading out arriving information from the server side. The Virtual Office had a Java Applet such as TeamPortal [HW] already running in their environment and this could just be wrapped as a subclass of the AwareDB class. The Java Applet was included in a frame of the Virtual Office website to connect to its system. As a result, the communication system of AwareDB was a part of the whole system, and the Applet of the Virtual Office mediated the communication to AwareDB on the server side. To make the application a subclass of AwareDB, the following commands were inserted into the application:

```java
// import package aware which is of AwareDB
import aware.*;

// declare the class of application as subclass of ClientApplet
public class VirtualOffice extends ClientApplet {...}
```

The mostly used methods by Virtual Office applet are illustrated in the following table:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect</td>
<td>There are several connect methods with different input parameters. This one serves to connect a user to the application and then to AwareDB. It is also used to control whether the user’s connection is still alive or not.</td>
</tr>
<tr>
<td>publish</td>
<td>This method is a general connection method from Java to JavaScript. The Virtual Office was mainly using JavaScript for the graphical user interface. The application interface consisted of several HTML-frames. What to publish (content) and where to publish (the name of the frame) are sent as parameters to this method. As a result, the interface gets refreshed with new content. The name of the frame on the interface is defined by the developer and has to be managed accordingly by the JavaScript side.</td>
</tr>
<tr>
<td>process</td>
<td>The method process gets the result of the queries from the database back. This is also where all information arrives. Thus, the result has to be received inside of this method and the next process using the result should be programmed depending on the application.</td>
</tr>
<tr>
<td>send</td>
<td>To send a Package object containing any information to any one.</td>
</tr>
</tbody>
</table>

As mentioned before, the discrepancy between the tools of the two systems brought difficulties with being aware of the activities of the other system side. An additional object called `Query` was created, and it was added in a package as the content for sending the activity of one side to another side. Using these methods, Virtual Office could cover the process of connecting to AwareDB, getting information and delivering information to each specific user.
• Run a script to start up AwareDB with the right database description. AwareDB connects to a database using JDBC. The information about which database or who is the owner is not fixed and has to be declared accordingly to the application. For this, a template script was given that could be changed easily with their database name and managed the security in the developer’s file system.

Mediator is the class of AwareDB that has to be started first. It has to be started on the server on which the database is installed. This script contains information about the server, the database name, and the class path as shown as follows:

```
jre -Dname=DBMS_NAME -Dsession_server='HOST_NAME'
    -Dsession_db='DATABASE_SID'
    -cp NECESSARY_CLASSES_PATHS Mediator &
```

This short script contains the minimum inputs. When starting the Mediator, the report appears as follows:

```
Mediator STARTS
    connecting to the database.....
    writing the port......
    DBMediator for mysql successful
    Wrote port number 5314 for mysqlMediator into /home/case/.port
    And also table metadata is created
```

The designer’s work steps to integrate the AwareDB base system structure and the Virtual Office application are described. This was achieved with the help of small wrapping scripts provided to Virtual Office and an extra object class implementation on the side of AwareDB, without touching the implementation body.

**Adding functionalities for users’ interest**

For the second step of the integration process, a specific part of the application Virtual Office needed to be considered. This functionality is called “Holiday Planner” and basically lets members define whether they will be working or on holiday through the environment. In case they mark off days, they can precisely record what kind of holiday it will be. For example, it can be real vacation, or they might be attending a conference, etc. This kind of calendar is not only information storage, but also a sensor that can inform about the availability and workload of a person in a group. It is to be taken seriously with regard to the problems associated with it including the violation of privacy and the monitoring of the attendance of a person [MT01, Pal99]. This Holiday Planner, however, does not inform people about others’ holiday schedule automatically in any way, nor does it let people describe in what kind of data people are interested in with respect to another person’s holiday.

With these goals in mind, three graphical interfaces corresponding to the three
Figure 5.12: Holiday Planner: a part functionality of the Virtual Office allowing people to declare their vacation time and to find out that of their Coworkers.

windows of AwareDB functionalities were provided, see figure 4.1, figure 4.2 and figure 4.4 in Chapter 4. These AwareDB functionalities can be included in an application by adding a link. Let us say the interfaces of AwareDB are called Interface1, Interface2, and Interface3. The following example shows how one of the windows Interface1 could be called by a JavaScript function, which might be on the side of the Holiday Planner:

```javascript
function openInterface1(userid) {
    F1 = window.open("../Interface1?userid=userid","modeller",
                    "height=700, resizable=yes");
}
```

First, the functionality of AwareDB, provided as a directory, was saved in a web accessible place of the file system. The necessary part is only the path leading to the provided applications, “../Interface1” and the user’s identity, such as “userid”. The others are optional depending on how the developer would name and represent it. These include the function name “openInterface1” and the size of the interface, 700.

As a result of the integration using the aforementioned JavaScript on the Virtual Office side, three functionalities of AwareDB were offered. The first functionality is a window to allow the developer to declare certain data entities visible to the
members (see figure 5.13).

<table>
<thead>
<tr>
<th>computers</th>
<th>group_task</th>
<th>group_user</th>
<th>groups</th>
<th>messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>project_group</td>
<td>projects</td>
<td>rooms</td>
<td>tasks</td>
<td>userSettings</td>
</tr>
<tr>
<td>user_task</td>
<td>users</td>
<td>vacations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tables read out from the database at the moment, "vacations" is chosen

**Entity**

<table>
<thead>
<tr>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
</tbody>
</table>

Attributes of the entity "vacations":
- Selected one will be defined as visible to users
- Shown only attributes made by the developer
- These are virtual attributes added by user
- To create a new virtual attribute

**Visible Attributes Now -->**
- timecode
- vacationType

Figure 5.13: Developer’s tool to manage data. It especially allows the developer to choose which data attributes are made visible to the user. From that moment on, the user can define their interest related to the actions being executed on the attribute.

Compared to the prototype version, it allows the developers to create additional attributes to a data entity (see also figure 4.1 in Chapter 4). These are nonexistent attributes of entities in the database schema and thus, named as virtual attributes. The reason why this functionality was introduced is that the developer of Virtual Office was strongly concerned with the graphical data representation on the whole and felt uncomfortable or found it sometimes meaningless to show the data one-to-one as it is. In this way, the developer could compensate the difference of the form of the data source and of the interface.

The second functionality allowed users to declare their interest about other users’ holidays (see figure 5.14). Compared to the original prototype, the members of Virtual Office could have more choices for the manner of distribution, for example, via email. Here, any other possibilities, such as SMS, could have easily been added.

The third functionality, which automatically creates an evaluation page following each change, was to show how well the designer’s initialization and the user’s perspectives met each other. Also, the evaluation shows the visible data table and the number of members who are interested in all kinds of information change. Six of 51 members were interested in being informed in case the attribute, timecode, of entity...
The preferred manner of information delivery

Table 5.1: Developers can define whether any action has to be taken by the system in order to inform about database events in relation to specific attributes or entities.

Users can define whether any action has to be taken by the system in order to inform about database events in relation to specific attributes or entities.

The preferred manner of information delivery

Figure 5.14: Developer’s setting for system behavior: to define how the system has to react with regard to different kinds of data transactions. A similar interface is provided to each user.

vacations was changed.

To sum up, the integration was made by wrapping the application in a Java applet and adding AwareDB functionalities for modeling interest as HTML links somewhere in their application. Otherwise, there were changes on the AwareDB site to allow for a more efficient integration.

5.3.5 Result and Interpretation

We accompanied the whole process of adaptation to observe and help the developer use the prototype. Benjamin Stäger, the designer and the developer of Virtual Office, participated in this evaluation by completing the integration process and answering given questions, see Appendix. In addition, some users commented spontaneously on the resulting application.

The effort to explain the context was quite low. To explain how to use the system to a developer was, in fact, the easiest part. The reason was, because the developer was aware of the problem in his application and was looking toward the system to help solve it. For example, the application, Holiday Planner, was chosen, which was in development and lacked aspects of the system. From the developer’s view, it was a known scenario and, thus, easier to explain why AwareDB could improve the situation. On the other hand, it was less successful in explaining by using a short documentation, or user’s guide. First of all, a short description containing the steps necessary to run their application with the system was proposed. But the developer claimed he preferred to first look at the source code to see how it worked. Many developers have different work styles. The developer of Virtual Office was someone who understands others’ work mostly through their codes. Thus, the of-
The number of users who are interested in getting information about events according to the actions.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Action</th>
<th>Setting</th>
<th>User nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>vacations</td>
<td>insert</td>
<td>do nothing</td>
<td>3/51</td>
</tr>
<tr>
<td></td>
<td>update</td>
<td>refer to personal setting</td>
<td>5/51</td>
</tr>
<tr>
<td></td>
<td>vacationType</td>
<td>refer to personal setting</td>
<td>5/51</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>do nothing</td>
<td>3/51</td>
</tr>
</tbody>
</table>

This evaluation shows the case of the entity "vacations".

Insert, update and delete are declared as possible actions that could happen to data.

Actual setting of the system according to the actions.

The number of users who are interested in getting information about events according to the actions.

Figure 5.15: Evaluation showing users’ and the application designer’s perspectives on the application data.

ferred documentation, containing a small explanation about what it is for and the steps to do to integrate the system, was not very helpful for this developer but may be more helpful for others.

To adapt in detail, such as adjusting the code to the graphical interface needs, took more time. All together, it took around four days for the whole case study. It is, however, useful to note that the most time consuming part was adapting AwareDB to the unfamiliar system environment. Typical problems included incompatible Java versions, its installation on different machines, or incorrect class paths. The application designer suffered with the Java class path [Inc], something is not easy for casual users to work with. This seems to be a very familiar problem of Java users who have to run an application. In the end, however, AwareDB was successfully integrated into the test scenario, Holiday Planner.

The developer commented,

"... another advantage using the system is that the [application] designer can cope most accruing/arising tasks using only one application ..."

In this case study, the system was very much appropriate for the developer’s needs. The communication system was one fundamental part of the system that could be used in several functionalities in the whole application. These applications included message sending, chatting, and presence notification.

The developer also commented,

"The new applet is intrinsically smaller than the one I have used before, and the data transaction to the server and those overloading were reduced."

102
We have seen how the developer had already tried to substantiate similar possibilities in his application, when he started to recognize the problematic of communication. He said he had to assemble more than three code files to meet the same requirements using his scripting languages. The argument of the developer assured the system quality with regard to the database transaction too. The most visible and immediate effect using the system was that the developer’s work for making the system automatically inform the users was reduced substantially. The class “Package” of the system easily helps to manipulate the data receiver and sender to enhance the whole information sent to every direction (see section 4.4.1).

The developer also commented,

“...the designer doesn’t need any more to take care of the tasks to inform users what new information has arrived in the system.”

And the users who got that information were saying,

“Hey, ...our database sent me an email saying that A will take a vacation ...

Of course, it is not exactly the database that is sending emails to people or delivering messages on the monitor immediately. However, the effect reached on the people was exactly that: The database is saying something, such as in the comment above. The effect was increased because nobody knew about the other’s interest. There were so to say, ongoing conversations between the user and the database, which were kept in privacy. Since it was not represented in the common way on the common interface, information was delivered faster on the whole:

“...our users had then shorter waiting times and got the information about new data faster than before.”

With regard to AwareDB, there were three windows offered by the system prototype. The adaptation for the second part, Adding functionalities for user’s interest, took more time than the integration of the communication system. This part of the process is analyzed as follows:

Showing the database to the final users as it is was not originally considered an option by the developer of the Virtual Office. This concept came additionally with the AwareDB functionality. The developer was not very comfortable with this at the beginning. However, the fact that he can control the visibility to some degree by deciding which part should be visible to the user led him to the decision to include this functionality. It was very soon understood as programming help, because the developer no longer needed to write as code the data that needed to be offered to users to allow them to describe their interest. This was simply done by defining certain entities and their attributes as visible. After all, this fact led to the idea to think about the comfort of visibility in the perspective of the developers.
The data representations differ between the application and the database side. There is data that the developer found necessary for delivering awareness information to the users. This is represented as one data object but is often not simply one object in the database. It is built up as one object out of the database query results. To solve this problem, a functionality allowing developers to define virtual attributes is added. Adding a new entity could, of course, be accomplished simply using a DBML (Database Manipulation Language). It then needs more work on the application side adding codes to build the processes that are required with regard to those new entities. Therefore, it is only allowed to add the (virtual) attributes based on existing data entities.

The possibilities open to the users using the system were reduced to describing their interest in a very limited way. To explain to them personally what they could do took a very short amount of time. Afterwards, there were positive and negative reactions about the visibility of the data structure from the user side. The positive ones were mostly based on their experience having messages sent about the database event. However, some people felt uncomfortable, because it was not like any application that they had used before. Instead, it was an additional module linked to their application. Also, a more general way to describe their interest was asked for, such as typing in a sentence.

5.3.6 Discussion and Summary

Based on the experience, the following points can be brought up for discussion:

Developer’s position

We experienced that the application designer of collaboration tools, including Internet-based ones, need to have a broad knowledge about the technical aspects. This was shown alone by the issue of connecting users in the application. In addition, they also have to be able to understand how to use the database systems. In fact, the applications are often a kind of mirror of the database, because the data content is often the main shared material connecting the users. Furthermore, it provides the impetus to collaborate. At the beginning, it was surprising why the work was not clearly subdivided and distributed among specialists for each part of the application. Unfortunately, it was observed that graphic designers, interaction designers, and database specialists very rarely work together. Secondly, this whole situation becomes even more hopeless when the specialists do not understand what the others are doing. Then, it was asked why not choose an existing graphical interface to the database, which even allows people a kind of visual programming using drag and drop. The reason seemed to be the integration factor. Otherwise, it is often the case that the developer misses the next step of integration, such as what to do next when the database is nicely designed?
Database’s position

In an application tool, a database is the central part, and yet its importance is often underestimated. It was often to hear application developers saying that they are trying to show the database content. They often consider the database as the center of action because all the data and changes are registered there. However, during the process of their work, they move the database more and more to the back of the scene. We are not arguing that this is wrong. We just would like to consider why this happens. Even though the graphical interface and database are assumed to be so strongly connected together, it happened again that there are not many given solutions for merging them together for developing groupware. Thus, this thesis argues that AwareDB is a step forward for giving a reasonable answer to this problem.

Transparent database

There were discussions at the beginning of the work as to what extent the database needed to be visible or if this was necessary at all. In this case study, it was chosen only to achieve one goal: How can we let people describe their interest themselves, with the idea of how can we let people feel that the database is acting for them. Positive experiences through this case study included successful data management in collaboration with the application designer and users. This can be further managed by increasing the responsibility of the users. However, the question “To what extent?” seems to be the responsibility of the application designer.

In summary, this second case study showed us that AwareDB could improve the Virtual Office application in the direction of an active and forthcoming system. To achieve this, functionalities were provided allowing the developer and users to declare their interest and showing the differences among developers expectation and users desire. This evaluation helped the developer and users to be aware of each other and stimulated a kind of collaboration among them. Especially the developer could modify the application so that it corresponds to the users’ expectations.

Virtual Office used functionalities provided by AwareDB with relatively little effort. Using a properly functioning communication system, it could move over from the passive to the active information transfer modus. It removed inconveniences with regard to the correctness of the information. It could also be forthcoming to its users with the information that they are indeed interested for.

AwareDB verified that it could support group applications to handle awareness information. The insight about the relation between application development and the position of the database was gained by accompanying the integration process in the case study. AwareDB was included as much as possible to reach the desired outcome.
by changing many details of its functionalities. The many small details, which had
to be changed for the integration, emphasized the importance of the early considera-
tion of the database in the application development phase. To support forthcoming
behavior, AwareDB enabled the use of the existing database design. However, it is
desirable that the database would have to be restructured so that it considers how
the application wants to provide a better awareness to the user about the data.
5.4 Support for general Information Visualization Concept

In the past few years, we became familiar with fashionable and sophisticated data representation. Increased Internet use and Internet-based development especially make it advantageous and even necessary to attract people to one’s own websites and web-based applications. According to these developments, old-fashioned information representation, such as the table or bar graphs, belong to the past. Considerations to represent data aesthetically and also meaningfully found their own position in the research world. New techniques are able to handle different expectations, including easier overviews, better navigation, fast content catch, and interactions independent of the dimensions in which the data is shown. Among these works, this thesis is interested in those with strong aesthetical aspects for the visual effect. Furthermore, the works were approached from a conceptual standpoint to reveal a general solution for the database itself and its content representation.

In this section, a conceptual work of aesthetic data representation will be introduced, and it will be explained how the forthcoming database concept is intended to support database use. The study focused on general solutions for intuition and interactivity in data representation, while the forthcoming database concept focused on analyzing the needs of the visual concept and finding solutions in data management support. As the final result, a system having a good balance of both aesthetic and technical aspects could be presented.

This chapter starts by giving a short overview about the basic concept of the Blue-C project and Infoticle application. Further, works related to the information visualization metaphor will be introduced to point out the focus of this case study. Then, an application will be introduced, which is built up in an immersive virtual reality project [Imm] and the specific metaphor “infoticle”, which was created by Andrew Vande Moere [Moe02a]. Subsequently, the applications representing financial data and web accessible files in an architectural office using infoticle will be described. The findings from these experiences have proven the necessity of the database support that broadens the Forthcoming Database context.

5.4.1 Blue-C and Infoticle Application

The blue-c project at the ETH Zürich aims at fundamental research in a virtual environment in which people can interact with 3D models and other people in a novel level of immersion. As a result, an advanced virtual reality system is expected due to the high quality of full immersion, multiple video streams, human inlays and sound rendering.

To develop applications that improve the usability and performance of this blue-c
environment, collaboration with information architects was needed. For the group, CAAD and Architecture, the blue-c space created by three projection walls (see in blue-c [Imm] → concept → cave) was an opportunity to realize an “information interaction space”, in which data can be visually represented and people can interact with it for refinement and reevaluation. As further motives, the CAAD and Architecture group wanted to increase the seriousness and usefulness of the application, which was generally said to have failed in conventional aspects, and to let users experience the data not as a static, but rather “lively” representation.

To this aim, an application was developed that was based on two main work directions: aesthetical and technical. The first one resulted in the establishment of a scene. It is the interface to the user and contains all data representation and interaction possibilities. The second one comprises all of the procedures behind the scene. These are responsible for providing data from the database to the scene and delivering the response according to the users’ interaction with data as the scene update. Thus, the Infoticle concept and forthcoming database were offered as an application system, which demonstrated a new usage of the 3D environment. Even though the application is built in a 3D immersive virtual reality project environment, it is operative in 2D as well.

5.4.2 Artistic Representation and Explorative Data Analysis

There are works focused on artistic values or on principles of visualization. However, works founded on a general metaphor of a visualization concept combined with aesthetic values as well are very rare. To behold the data visualization tendency, the work were grouped into two topics: “Artistic and Intuitive Data Representation” and “Explorative Data Analysis Capability”.

An aesthetic experience is a kind of intuitive experience. A system that is alleged to be intuitive should also be aesthetic. The system “The Living Web” [LGSM02] which is the CAVE version of the “The Living Room”, shown in Figure 5.16, gives us a sense of comfort provided by an eye-pleasing environment. It represents web data, especially image and sound information, in an immersive 3D environment. Some key words, which are gained from users’ conversation inside of the space, are used to search related images on the Internet. The resulting images are projected on up to six surrounding screens and offered as interaction possibilities to users, for example, to grab an image to get more feedback about it. The main goal of this system is to offer users an intuitive interaction comfort, which immediately results in an improved ability for them to demonstrate their individual preferences. Compared to data presentation using charts or bar graphs, the Living Web environment looks very attractive for users and intensifies users’ pleasant and intuitive interaction by taking a specific data format or “image”. In addition, this could be useful to apply
for a large database of images. However, it is hardly imaginable to have images as a general form to replace all kinds of data, since an image often just represents the partial content of a document or part of an image bundle.

To use a cube or a circle as a metaphor for data representation might sound less attractive or electrifying. In spite of such biased opinions, the cubes of the ICCS project, shown in the figure 5.17, visualize the aesthetic navigation environment of a database containing meta information of work documents organized in a file system hierarchy. This was realized using the maximum technical possibility at the time of development, as it was implemented as a 3D Web “navigation delegate” that allows user interaction and immediate result representation based on a real database connection behind it all. It was really an aesthetical result of using cubes for managing documents accessed by members in different locations, but there were critiques such as how the user could lose their orientation when they were inside the cubes, even though functions such as jumping outside of the cubes were provided. For users who

Figure 5.16: “The Living Web” in the CAVE environment, by Lopez-Gulliver [LGSM02], page 12

Figure 5.17: Cubes Delegate: different view during navigation in a document database, see in ICCS project’s homepage [Too], Delegates → Cubes Delegate

do not know what to look for, a large database makes it difficult for them to get desired data or to find out hidden knowledge in the database. VisDB [KLTW95, KK94]
shown in the figure 5.18 is a system for mining large databases. The basic idea of this system is to include the human ability in the data mining process and *visualize as many data items as possible at the same time* by mapping each data values to a pixel and arranging them adequately. This system is based on a query-feedback process and considers how to give more feedback in providing data also with “less” correlation to the result. At the beginning, a user makes a query to start the process that might make the resulting data already selective among the whole data sets. The results are then displayed in many separate windows using pixels. They show different arrangements, patterns or distances from the middle by mapping the relevance of the result to colors. The user then can explore the result and he is expected to refine his query. Unfortunately, this system is capable of almost overloading the user by offering so many windows of query and results. Also, an intuitive media to set a query is not given in this system, for example a mouse click on a pixel. Users have to know how to set the query using a specific kind of notation.

![Figure 5.18: Basic visualization method of the system VisDB. The resulting data items of a query are sorted according to their relevance with respect to the query and the relevance factors are mapped to colors. Finally, they are arranged on the screen from the middle. Each window shows different dimensions. See in [KK96]](image)

Other similar visualization techniques for database exploration, such as circle segments, exist [AKK96]. This technique proposes to visualize large amounts of high-dimensional data using color pixel per data value.

The resulting observations could be summarized as follows. First of all, the research situation seems to be giving more effort to make functional applications rather than to find a general concept using the full advantages of specific environments, such as immersive or 3D environments. Also, the consideration to include aesthetic values in functionality-oriented scientific applications seems to be very much a luxury, even though one of the main goals of the visualization is to enhance usability, which is related to the aesthetic, or so-called *beautiful interaction* [HO00]. Secondly, the typical concept and applications related to the immersive world are focusing more on
huge data sets and simulation including fluid flow, chemistry and biology data. In such work, the consideration of real data and real database connections is negligible. The common impressions of these visualizations are in one word “impressive”, not only for their visual effect but also in the size of the resulting system. It can be justified that their goal is to make ubiquitous what is now available, gathering all possible best technologies in a system. These are huge and usually comprise a collage of different research domains. However, here again, aesthetic considerations related to the interaction are lacking and real data with real database connections is hard to find compared to the file system used. The third observed point is that such visualizations assume that the users are familiar with the content of the database. Otherwise, most of the abstract visualizations, such as using lines and points, have no chance of being understood by users.

5.4.3 Infoticle, Time Varying Data Visualization Metaphor

Vande Moere worked on a new information visualization principle [Moe02b, MM03, MMG04] for work spaces based on 2D as well as 3D user interfaces. The main goal of his work was to provide users intuitive and interactive data exploration possibilities. The main effort was given on merging both visualization and interaction techniques. As a result, he coined the term “infoticle”, which forms the general fundamental element of his concept embodying information.

Figure 5.19: Infoticle scenes: (a) A user immersed in an infoticle system (b) Data sources of students and money are spreading out in the scene (c) Density view of infoticles

The infoticle system uses query-independent visualization to sort out the data according to one dimension (e.g. time order) and uses the metaphor infoticles as a screen-filling pattern to spread out the data on the display. In case there is no natural ordering of the data, it adds some kind of time order to the database. Even though it is using time arrangement, the data is controlled by Newtonian-rules, in an attempt to be useful and more interesting for interactive exploration. Each in-
foticle object can be one of three states; born, alive and dying. Each data object is instantiated by an infoticle and its life procedure is controlled by Newtonian rules. As long as an infoticle is alive, it is specially used by users’ interactions including pooling, pointing, and defining a value. The infoticles influenced by user interaction return information back through their behaviors such as clustering. The infoticle environment additionally provides tools for user interaction including emitters, forces and surfaces in the space. An emitter can be compared to a fountain and presents a class of data. Thus all of the data of a certain class is poured out of an emitter according to one dimension, for example, time order into the 2D/3D space in the form of infoticle streams, which characterize their birth. Users can have hardware help such as Flock of Birds to set values of forces and surfaces. In case the values that are given are predefined, users can select the value by clicking one of the surfaces in the space. These values inspire the behavior of infoticles. Flying infoticles can gain or lose speed, pass or be bounced at a surface and build clusters in the scene (see different kinds of clusters in [Moe04a] → C.Color → Galaxy World Patterns).

The scene can also be frozen by the user to allow an overview of his or her data refinement and exploration processes.

5.4.4 Challenges and Requirements

Challenges of the infoticle

In addition to the advantages using a 3D immersive environment, including obtaining qualitative and quantitative information, an infoticle-filled space still proposes a number of challenges:

- **Generality**: An infoticle can be any data, an image or ordinary file, so it does not show only a certain kind of data. Also the dimensionality of the data is not limited.

- **Capacity and Performance**: The space can contain a large amount of data with surprisingly fast-moving infoticles.

- **Diversity of tools**: There are different kind of tools provided to users. The diversity allows users to formulate different queries and to use it for the analysis of data.

- **Intuitiveness and Immediacy**: The space with infoticles attracts people to start any action, and tools offered in the space help them for further exploration with the data. The influence of users’ interaction is immediately visible.

- **Autonomous and pleasant**: Once the data is born in the space, they move around with help of the Newtonian rules. It is important to clarify that the system did not generate infoticles according to a specific task or expected analysis. Thus, a user could just stay in the space and watch and enjoy seeing how
the life of data sets in the database evolve, which is a pleasant fact.

Requirements for the Data Management Support

During this project, different aspects of database support needs were faced because of the generality of the data metaphor and specific environmental needs such as interaction. Those included the necessity of a general structure according to the metaphor, including the high-speed answerback and flexible preparations of the data before data requests. Considering the experiences and above challenges of the infoticle concept, applications supporting data management were built. Some trials were performed according to the technical requirements working with the application using infoticle.

- **Data exploration through open-minded collaboration**
  Even though a visualization space can be used by multiple users, the connections to different domains broaden the user circle and so, offer a widened collaboration environment.
  Usually a 3D immersive environment is quite isolated because of the hardware and software limitations and thus, the possibilities of collaboration among users can be discouraged. The blue-c project, for example, set as goal to build at least two caves. One of the main characteristics of such an environment is to provide it as a multi-user system. Thus, it should also work as a collaboration environment, not only as a demo space. For that, basically, it is important to have a connection to any other domain, because it will increase the possibilities in many aspects, including discovering knowledge and problem analysis. The connections of different domains will easily help people to join the world and overcome the barrier to the 3D isolated world.

- **Support general visualization paradigm**
  Infoticle is a visual embodiment concept for “all” kinds of data objects and it is important to address all data entities as if they are objects of the same class. In contrast, the database is usually redesigned depending on the needs of a specific application and every application could have different dimensions. To match both a high-level interface and an underlying data model, a general structure of data sets might need to be invented.

- **Voluminous attribute**
  It is common that data attributes contain numeric information as values. To enhance the visual interpretation of proportionality, it was often requested to provide numeric values translated into objects even though they were not considered as entities in the relational database. For example, the number of students of the department of architecture is 1000. Instead of representing this simple value 1000 as a number, producing 1000 objects of “architecture
department” and representing it in the scene will offer much more visual information to the user. The possibility to handle data in relation to volumes of objects converted from attribute values is more required in an immersive 3D environment in general (different volume than tomographic volume [TSH98]), not only in the infoticle application. This might harm the image of data visualization according to the generality of its concept. However, this can improve the system on the visualization level much more.

- **High-performance access to secondary storage**
  The requirement of data delivering performance is poor in case one uses only the direct connections to the database. To show a large database, just the transactions to retrieve data sets will take forever so that an interaction is not possible at all from the very beginning. This is a very common problem in scientific computation including large model simulations or medical scan systems [BSS+01]. Especially the rendering performance gets crucial if the interaction should take place in a simulation, because the data which is pre-calculated and saved does not have much chance to react to an interaction and feed the data for quasi real-time rendering.

- **Reproducible results**
  Each moment of the interaction processes while a user explores the data in an environment should be reproducible. This might be a snapshot or movie. Important is that the user can show each moment of interesting knowledge findings to others or review them after his or her interaction. Even more preferable is to reproduce the scene and all the visualization so that the user can continue his interaction further. However, it is very difficult to save a scene in the form of an underlying data model back to the database. It will also cost too much compared to the result. But a history of interaction processes, for example describing the positions of the important elements in the scene, would be reasonable and might be enough to reproduce the scene again. A work of Reed [RGC97], page 42, cites, “Many studies have shown that recordings of results and histories are key components in collaborative data analysis and user information internalization.”

Using the infoticle concept and the prototype AwareDB, the connection between the blue-c application environment and the Web-based application environment was implemented. It was to enable the opportunity that two kinds of domains communicate, send data to each other and make the other side visible. Secondly, the capacity of the visualization and the framework side using a simple table was tested. The purpose was mainly to produce as much infoticles as possible, to observe how the communication between the framework and visualization work well and to test the power of infoticles with respect to how many objects can be run in the space without having speed problems. It used only two very simple tables having around 25 rows. Finally, an application for an architectural company was made that has
a large database archiving all of the web access logs. Any access on their website was registered in the database, which included the classification of the subject and object. Thus, the database was growing at every moment. They were curious to see what kinds of data their coworkers mostly used.

5.4.5 Connecting blue-c to the Internet World

The blue-c portal with three projection walls was of an isolated nature. It was physically built on space and was planned to have only one connection to the other CAVE environment, which provides identical technical possibilities. Other collaboration areas, such as Internet-based applications, were not considered. Using AwareDB, two different application environments were connected: 3D immersive and an Internet-based application environment. It was clear to us that 3D VR can be used easily as one of the nodes in a connected collaborative world.

AwareDB, the framework prototype, was implemented in Java and contained a communication system to deliver the collaboration data among users in a confident and smooth way in an Internet-based application. In contrast, the applications based on the blue-c environment should be implemented in C++, because the blue-c constellation and its Application Programming Interface [NLSG03] run on SGI IRIX. However, it had neither Internet nor database access possibilities. To connect both worlds of AwareDB and infoticle, which were of different programming domains, the framework was first extended by added module to accept C++ language communication and implemented classes on the infoticle application side as well.

Figure 5.20: A module handling specific domains for the data transport to different kinds of application environments is added to the framework. In this case study, a C++ specific module was implemented.
module on the AwareDB side offers an open connection, which can be demanded by the infoticle application side. Once the connection is established, the AwareDB side can also supply the database connection so that the application stays coupled. By the signature kind of information attached in the delivered package (see also in the chapter 4, Information transfer protocol - package), the framework side can recognize the demanders’ domain and user’s identity. Based on this information, all the data sent and received will be handled by a specific module where the data will be translated into a transportable and suitable form to both domain sides.

The visualization application can also directly connect to a database, which is in turn connected to the database mediator instead of through the application domain handler (see in chapter 3 → Application Middleware System with Layers). However, the latter case can only be used as a stand-alone application. It can also connect to both AwareDB framework and the database and choose one connection for the database queries. In case the database queries are sent to the AwareDB framework, the actions and events happening in the immersive environment can be also available for the web-based representation and visa versa. To show this possibility of the two connected worlds, a web application that was already done in early studies was used. Having the connection to the blue-c environment, the information of the users’ presence with both sides could be exchanged. The communication between both sides, blue-c and AwareDB, was enabled using a package containing data and user identity. The package was received as data type “string” and streamed to each side. However, its representation was left to the application designer who would realize the interface.

5.4.6 Supporting interpreted Data Object for Financial Data

To demonstrate the power of the infoticle concept, a subset of early ETH Zurich financial budget data was used. There were two tables in the database, one representing departments and the other students. The former one had four attributes including id, name, family name and expense. The latter one had three attributes: id, name, and department. The application designer mainly wanted to show the “cost” of each department and the “student” objects on the scene. The rows of department tables were thrown out from a kind of source and started to move in the environment as infoticles. All the infoticles representing either money or a student of each department streamed out from their sources accordingly (see in figure 5.19, (b)).

At the first moment, the tables were read out one after another, first the department and then the student table. But the visual representation of the database was disappointing: Seeing all of the students coming out and then the money is visually not very interesting. We usually understand a database as a container for different kinds of data well organized in some sort of form. However, some visual representations, in turn, need to access all data at once. It is as if a person opens a jar and sees all
the contents at once. For this case, the tables of the database should be read out simultaneously. To do so, a database connection was implemented for each emitter. Whenever an emitter was created, a new database connection was established. In this way, the emitters were only responsible for a certain class of data, providing data in the environment relatively quickly.

Then, it was requested that the costs be spread out of the source even though they represented a value of an attribute, not an object. However, the application designer asked to have the attribute values take the form of application objects and initialize them as infoticles. It would be advantageous to facilitate a qualitative rather than a purely quantitative comparison of the resulting infoticle clusters. From the visualization, it is understandable to ask any other form of a number, for example, “100”. People will not gain the volume if they see a number “100” swimming far off in the horizon. Rather, one can gain an understanding for the volume by seeing a cluster of 100 points or balls. Also, with the specific characteristic of 3D environments with equipment available today, it is easier to interact with 100 balls compared to doing division with the number “100” and other calculations. Thus, for infoticle representation, the designer wanted to have the coasts as infoticles, i.e. objects. This is referred to “Interpreted Data” in this thesis. As a solution, a class containing meta data information and methods to retrieve the interpreted data separately were provided. Through this class, the application developer can define which should be handled as interpreted data objects in the application. Once this class is loaded, the data object of this class is handled like other normal objects retrieved from the database, and all methods made for interpreted data are activated. A private method of this class:

```
setItp(Name, Parent, Attribute Name)
```

creates an interpreted data class “Name”, which is actually an attribute of the class “Parent”. Calling a method `setItp(“money”, “department”, “budget”)` will create a new data entity “money” which is an attribute of the data entity “department”. This will assure that the application handles the money like a normal object. In addition, calling the method `get (“money”, 4)` will produce a certain amount of money objects of value 4. The number of objects is certainly dependent of the original value of the data where the money object is an attribute. The distinction between a normal infoticle and an interpreted infoticle lies in their transformability in the scene. A normal infoticle object can be turned into a query-interaction tool, for example, a filter. The value of one of its attribute can be set as a filter value. That can influence the entire behavior of other infoticles in the environment, but this cannot be allowed for the interpreted object.

We also discussed about the option of creating and saving the interpreted data in the database before it is read out into the 3D scene. However, it was decided to put this process on the application side. This decision brought the advantage of better performance and also increased flexibility of the application development process.
The developer can change the values of the interpreted data whenever they want to rapidly provide other data exploration possibilities.

As a result, it was enabled to observe the effects of using an infoticle system that can simultaneously handle 12,000 data objects on the scene with a minimal installation of the infoticle metaphor method.

5.4.7 High Speed Demand for large amount of data stream

The infoticle metaphor gained the interest of company Arup R&D resident in London. This company has an Intranet storing the company experts and accessed through the Web or locally by their co-workers from 32 different countries. Any access on their data is registered using IP numbers, date and time. However, the people in this company were dissatisfied by the monthly log reports. It was not sufficient to answer the questions including the geographic regions that could access the documents and which special documents could be accessed by which people and with what expected relevant personal skills. The actual log report was so frustrating to them that they began questioning the correctness of the log data. Using the infoticle system, it was able to visualize the log file of one month with reasonable performance, which meant 128,000 log entries generated by 4654 unique files (see Section 4.3 in [Moe04b]), even though additional work of the application designer was necessary to convert the log files to a readable database table. This work was done in collaboration with two researchers sitting in different locations in Europe, both programming on Linux machines and communicating only through voice conference and emails.

“Faster, MUCH faster”. These were the most feared words as database application developer during the whole development process, especially for someone who could only work on a Linux desktop machine. It was more stressful than all other kinds of technical conventions and barriers. The speed is seriously connected to the aesthetic and interaction comfort of an environment. Nobody wants to see flying data suddenly freeze and at the very next moment move further along. This asks users to consider the freezing moment and disturbs the fluent interaction. Of course, this problem could be solved if better machines and higher connection speeds could be used. But, two Linux machines were used for the whole development. One could handle parallel processes and the other, not. To show every object at all, one sends a query saying, `select * from table_name;` But in such a log database such as the one at Arup R&D, simply reading out each object could take forever and would maybe not even be possible, since the list is growing at any one moment. Also, other implementation strategies, including changing char* values to index integer numbers, were experimented with.

The core of the infoticle metaphor relies on motion characteristics to convey in-
formational values. Therefore, rendered infoticle animations should never be interrupted by computing expensive algorithms such as database calls or data look-up processes. Thus, to guarantee high interaction performance in the scene, parallel processing and shared memory organization were incorporated. The data cache was located in a shared memory area and could be accessed by both sides: the visualization application process and the data handling process. Both processes were in a master-slave relation. Whenever the application side signaled the data supply, the data retrieval process was activated. However, the synchronization accessing the shared memory was done through programming. Another functionality was added to the Cache module. A process that could be evoked by the acquisition process was included to prepare the necessary data in advance. Since the data supply was asked for per time stamp, for example one hour, a rubric was used in the infoticle metaphor. It was also possible to prepare certain amounts of data beforehand. However, the decision how big the time stamp should be was left to the application designer in the framework. In this way, it could be avoid slowing down the process or that the application process needs to wait for the data required, since the data is always prepared before it asks for “more” data. After all these efforts, the speed of data providing could still be improved. It might also be helpful if the database could handle parallel accesses, a detail which is not self-evident. Some projects could enhance their performance by stripping away the transaction management capabilities found in traditional databases [LJD97], an observation which delights us, because it implies that another concept of database architecture might be necessary.

5.4.8 Summary

The infoticle concept investigated how to present data to the user as living objects with respect to its time-varying aspect. It was valuable to apply this to the frame-
work because it is a “general” metaphor, not an application of specific design and finally could promote the framework in terms of generality. Also, it is conceptually highly appreciated, because the infoticle concept points out important but underestimated time varying data characteristics.

The framework of this thesis strengthened the forthcoming aspect due to the speed the concept needed for its smooth data representation and interaction. This was made possible by the process of preparing data beforehand. To support such a general concept such as infoticle, which makes the data come alive on the screen or in a 3D virtual reality environment, the middleware connecting the application and underlying system has to also offer a general concept for improved integration, while still guaranteeing the high modality of the system with respect to a wide range of visualization applications.

3D combines more data sets into one single scene [SvBE00]; each moment of user-data interaction should be recognized, and each interaction should evoke a quantity of user queries that goes back to the database. For 3D applications, however, files are replacing the data resource in many cases, since it is more used for simulations at the moment. Virtual reality technology and database technologies together are relatively unexplored territories in spite of their popularity. As such, the task with Arup R&D was a kind of evaluation where our framework was able to help develop a real usable and useful application with acceptable performance. Having a general visualization concept such as infoticle, it was able to pinpoint the places where the database systems have to be further improved so that the general data visualization can be effectively used as a tool for collaboration. This applies to the fast cache, general data structure, history management, etc. As Reed [RGC97] already emphasized, a high-performance access to secondary storage is a proven and important fact.

Again, we are convinced of the importance of having all kinds of connections available, including between different domains and different users. The connections become more and more a necessity for an application to be up-to-date and also to be open for development. The web-based environment is increasingly being used as a delivery mechanism for visualization as well as a source of information. It is also reasonable to imagine that such an environment such as blue-c with many limiting hardware and software constellations can be connected to other domain worlds to use it to full capacity. The web is definitely providing an attractive development path, but it could be still a luxury due to the performance requirements [RE97].
5.5 Analysis and Summary

According to the strategy in section 5.1.3 and the vision of the forthcoming database in chapter 3, it will now be attempted to evaluate the framework’s performance in the context of its successes, problems and further possible extensions.

<table>
<thead>
<tr>
<th>Case 1: Programming course</th>
<th>Case 2: Virtual office</th>
<th>Case 3: Infoticle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Web-based work environment for students in a lecture</td>
<td>Web-based office representing coworkers’ presence and accessibility in</td>
</tr>
<tr>
<td>Database</td>
<td>Oracle, MySQL</td>
<td>MySQL</td>
</tr>
<tr>
<td>Implementation Languages</td>
<td>Java, PHP, JavaScript, HTML</td>
<td>PHP, Java, JavaScript, HTML</td>
</tr>
<tr>
<td>Awareness in the interface</td>
<td>New events such as work update</td>
<td>Someone’s presence</td>
</tr>
<tr>
<td>Number of users in the environment</td>
<td>41 (20 groups)</td>
<td>51 (includes guests members)</td>
</tr>
<tr>
<td>Computer experience of users</td>
<td>mixed, mostly with knowledge on programming and databases</td>
<td>not known and not important</td>
</tr>
</tbody>
</table>

The case studies’ environments have been shortly summarized. It is difficult to find an opportunity to integrate an application framework with such high quality visualization concepts as with the case studies in this work. In collaboration with such application designers, insight about their work could be gained, complex problems could be addressed and an appropriate framework was conceptualized. Finally, it could be proved that this framework could make a standard database system into a forthcoming one, as defined in this work, to support visualization concepts in many aspects. In conclusion, more general facts including usability and utility will be
• **Generality**
  It was shown that our framework could be applied to different kinds of visualization applications. Using examples in which a variety of data domains using different concepts and technology were used, it was described which elements of the framework were available and potentially could be used. Web-based applications, such as the two case studies, are proving to be very popular work environments nowadays, and it was sufficient to find out the base requirements of those systems according to the architecture of such applications, including connectivity and awareness. Those requirements are defined as base components of the framework and guarantee the generality of the work. With the immersive environment application, the framework was extended and, so far, its general applicability as well. However, it is true that whenever a totally new task benefiting strong device dependency pops up, which can lead to poor system design [Ben92], it was necessary to do expensive adaptation in extending further functionalities.

• **Usability**
  In the concept of the framework, some facts related to the user and work of application designers are already taken into account (see interaction model in chapter 3). Assuming that the application designer is the user of the framework, the forthcoming database provides a tool for them, which is flexible and easy to learn. It is flexible in that the designer can remodel the application by accessing the data model whenever it is needed, for which a simple interface kind of program is provided. Otherwise, this thesis was relying very much on the designers’ and users’ opinion based evaluations. Since the application designers were one or two persons, the conversation naturally tended to revolve around the ease of use and learning and what kind of subjective experience had taken place. A positive experience is being able to recount comments such as “the database said...” or “now I can just distribute information to everybody without additional work.” When the interface was shown to the application designer where they can remodel the data for their implementation without using query language such as SQL, some kind of introductory explanation on how to use it was still needed. So the interface made by a non-visualization oriented designer is still cumbersome and less intuitive for the application designer.

• **Integration ability**
  The prototype is mainly programmed using Java and C++. The small interfaces of the prototype use JavaScript and PHP. For Web-based applications, it offers wide compatibility and is easily integrated. Since it targets a user class who is mainly concentrated on a visualization concept, the modules could overlap with those of the visualization part, depending on what the user needs...
and what is already built into their system. The choice which module of the framework could be used in practice is left to the application designer.

- **Adaptation cost**
  For different implementation tools that are used for visualization, the cost can vary quite a bit. Since the system was developed along with the application cases, much time was needed to adapt it to each application development domain. But web-based environments proved to cost less than the last case study in an immersive environment. For example, if just retrieving data is asked without using any remodeling of data for the application side, it is important to know the criteria for the preparation of the data beforehand. In this case, this is done through adaptation work modifying a class, although normally, this should be rewritten. This may engender additional costs for the developer depending on the programming quality and their knowledge about their database.

- **Utility**
  In all cases, a solution could be found by using the framework and matching their requirements, especially regarding the problems they faced. So far, the framework satisfied the utility of the system and only quality issues were left to discuss. It is understandable that better quality has no upper limit. For example, in the last case study, the application designer would appreciate to have a much faster data supply than this system could provide. A collection of many facts influences the framework performance including programming and technical hurdles. However, it might be possible to see few more scenes of the application on a Linux system that allows a faster loading of data.

- **Case 1: Programming course**
  Important requirements:
  *
  * Awareness enhancement in the course environment

  Provided:
  *
  * Change of the states and users’ actions related to work were shown
  * Automatic distribution of the information using Java application on the user side

- **Case 2: Virtual office**
  Important requirements:
  *
  * Communication problem

  Provided:
  *
  * Provide the communication model of the framework to extend their communication system
  * For the application designer, an overview of users’ usage in the system and data modeling interface using the original database model were provided
For the user, an interface was offered to the data model that allowed them to specify what they were interested in

- **Case 3: Infoticle**

  Important requirements:
  
  - Speed enhancement in retrieving data. Consideration how to provide different data using a general metaphor

  Provided:
  
  - Integration of new domain connection
  - Performance in providing data using “Preparation”
  - Application specific data formation, for example, data out of an attribute value, supported

Finally, the framework provided better transparency of the data supply in communication and modeling. This enhanced the application designers’ work and ultimately, user awareness while using the system.
Chapter 6

Conclusions

There is a need for a concept that integrates information visualization and data support. Although there already exist many tools providing various helpful functionalities, the role of visualization and the developer’s work to define and design an appropriate interface are often underestimated. Ultimately, these situation-specific tools are often overlooked by potential users due to the lack of an all-encompassing use concept.

Building an application to visualize information requires knowledge in different research areas, including information visualization, cognitive science, and database design. Among these, information architecture is a field in which architectural principles and aesthetics merge into computer science, a strong reason why it can make a substantial contribution to the area of information visualization. The difficulty in using existing tools to develop information visualization applications is that they do not consider the quality of the information architectures involved nor assure that this quality can be fully taken advantage of. However, it is a general understanding that aesthetic values increasingly influence user interaction with data and the usability of the system.

The current state of database development is centered on the idea that database systems can serve more and more as a kind of application platform. Two already existing database concepts include the active database and group activity database. However, the potential of the database systems as an active and collaborative partner still has not been realized.

6.1 Forthcoming Database Framework

This thesis proposes a concept demonstrating how the technical essentials for information visualization can be provided as an integrated platform system. The experience in collaborations with other researchers of the information architecture domain led this research work to define a concept to optimize the behavior of the
database in this context. This thesis documents how this concept could be successfully applied to different kinds of visualization concepts. As a result, many topics we mentioned in this thesis may be valuable for other research projects that require the forthcoming behavior of databases.

- **Database position**
  One of the main themes underlying this thesis is discovering the potential of the database systems for being a forthcoming database, or one that supports applications with its forthcoming behavior. Supposing that the database is one of the most essential components in the building of an application, we first moved the database position to the surface, where the interactions among user, designer and the applications take place. As a result, we could finally discover the database as an actor in an interaction and consider its forthcoming role as a realizable functionality.

- **Forthcoming as Functionality**
  Based on the results of our research into how the database took on its new role, a framework or forthcoming database was defined. Its architecture shows the functionalities of a database system, which upgrade it to being forthcoming.

  - **Preparation.** High speed of data feed is a necessity for a system in which a large amount of data is visualized. Users of these environments can greatly benefit from their interaction with fast moving data and enjoy smooth ongoing immediate feedback. The concept of this thesis proposes that data set preparation could be one of the main components of the forthcoming database to efficiently and quickly prepare data sets for visualization before they are requested by the visualization application.

  - **Awareness.** Information visualization applications already provide a certain degree of awareness information. However, providing more awareness tailored for the application concept increases the application designer’s work. He or she has to think about what the users have to be aware of and how to inform or distribute this information, etc. The forthcoming database treats users as active users, as described by Bannon [Ban91], and users are initiated in being involved in making decisions with regard to their interest. This eases the designers’ work in modeling awareness information as data objects to meet individual user needs and provides a predefined network for easy distribution.

  - **Transparency.** In information visualization, user-centered design became an important issue as well. To build a system in which the user influences system behavior, the forthcoming database offers a concept of transparency. Through interaction with the database schema, declared as transparent (or visible) by the application designer, users can express their interests and desires to a limited extent. Furthermore, the concept
offers a kind of evaluation showing what kind of expectations the users have.

– **Modeling.** Due to the difference between the visualization concept and the database schema, it is often not avoidable to change or transform the original data structure into the form of the application side. To facilitate the designer’s efforts, the forthcoming database provides a module allowing object modeling using an existing database schema.

– **Cache.** A well-organized cache offers the foundation for many characteristics of the forthcoming database. It serves as temporal storage for data sets or place for management processes to occur before the data is fed into the visualization scenes. It can also provide a shared area of data where more than two processes can serve the data stored on it.

– **Interconnections.** In information visualization, there are many possible and viable networks that have to be connected. These include the connections between users, user to database, and database to all of the users. To some extent, information transfer should be evoked within the system, not by the user. The forthcoming database defines a module to provide a base for this communication network.

• **Domain extension**
The Internet is understood as a new kind of application platform with its advantages in distribution and platform independency. Many of our case study applications were implemented and provided through the Web. However, there were some exceptions, such as the immersive virtual environment, where limitations were unavoidable with respect to the hardware and software. For this case, the forthcoming database was extended with a communication channel connecting both sides. It showed us the potential of making users of two different environments or applications aware of each other’s work and the results produced through user-data interactions.

• **Visualization concept**
Many examples of how information is finally represented on an interface show us a variety of possible concepts. Already in our case studies, we saw how they differed depending on the application designer’s idea and the characteristics of the data sets that were used. As such, how the application designers understand the working of the databases and what potentials of the databases are finally utilized differ as well. Some of them want to open the database like a jar and see everything at once, and others want to have only a small amount of data one piece after another. These diversities of personal perception were supported by the forthcoming database in different ways, case by case.

• **User-Designer Communication**
In applications including web-based lecture environments, the functionality of monitoring turned out to be important for being better aware of others’
actions. With monitoring, the lecturer could react to the users’ work before
the users recognized the problem. The forthcoming database added modules,
including an analysis view that showed the use of data or user interest on
certain data sets. It could be calculated and collected inside of the system
without direct work on the part of the user. This could also happen in a
communal work space and ask users from the collaboration. Throughout the
process, the communication between users and the application designer was
enhanced, and finally it also influenced the management of the application.

6.2 Forthcoming Database for Context-Awareness

Ubiquitous computing introduced a new computing epoch in which many computers
can serve one user anywhere in the world [Wei96]. It characterizes an environment
in which diverse electronic objects are well connected to provide services at anytime
and everywhere. In addition, tools of all kinds are allowed to connect spontaneously
to each other through integrated networks. In a world where a myriad of computa-
tional devices and the user’s feelings are calmly merged for convenient information
access, context, which refers to the physical as well as social situation [MD01], is
considered one of the most important factors. It thus serves as an interface between
the real and virtual world [Kir04]. For example, context can be the user’s physical
location or information displayed on a screen such as pictures or words in a tourist
guide program [ADA+97].

There are different approaches to achieve context-aware systems.
The system Gaia [RHR+02, RC03] is an application framework approach for pro-
viding mechanisms to construct, run or adapt existing applications to ubiquitous
computing environments. In an experimental physical space that contains multiple
devices and can sense user actions, the services built using the Gaia framework are
able to handle context-aware information with regard to each user’s situation. The
framework itself provides events evoked by new services or applications. The con-
text, as perceived by Gaia, includes people’s location, conditions within a room or
place, and the presence of resources such as software, application, service, device
and persons. All the information about soft and hardware entities in the space is
stored in the space repository. To do so, the framework uses components of a five
system architecture: all applications are described using its logic, state and func-
tionalities (model), the models are presented in different forms (presentation), the
state of context is managed (controller), the model and the controller elements are
coordinated (adapter) and finally, the entire interplay among components is man-
aged (coordinator).
The Context Toolkit [SDA99, DAS99] aims to provide a general architecture for
making or upgrading context-aware applications. Its general mechanism simplifies
the access to existing context information and the adding of new context informa-
tion from applications. As a result, a context-aware application built using Context
Toolkit should be optimized in the aspects of flexibility and compatibility. In this concept, the context is defined as any information that can be used to characterize the situation of an entity including persons, places, or physical or computational objects. The Context Toolkit manages multiple input sources in context-aware applications, allows additional abstractions of information, and supports the execution of an application by independent components. To achieve these mechanisms, it contains components such as widgets, aggregators, and interpreters. Widgets encapsulate information regarding individual aspects of context such as activity, while aggregators act as gateways between applications and widgets, and context interpreters translate context information into other formats.

Both Gaia and Context Toolkit provide a general base for building context-aware applications using different focuses and starting points in their mechanisms; the former is based on models describing each application and the latter is based on widgets encapsulating a context.

All the case studies in this thesis were concerned with providing “appropriate” information for users, in other words, information relevant to the context-aware computing concept. The visualization applications were supported by database systems, especially for user activities in specifying their interest according to specific data.

In the Virtual Office case study, effort was made to integrate user interest in order to strengthen the forthcoming characteristics of the application. In the main application provided in the Virtual Office system, the user’s location was taken as one of the most important pieces of context information. However, it is conceivable that location information can be shared by other application, such as a discussion board, providing tools to users that take into consideration the available display tools in a user’s location.

The findings of the last Infoticle case study of this thesis offer us the basis for many scenarios to examine in relation to ubiquitous and pervasive computing. As an example, one scenario might focus on hardware recognition. During this research, the isolated nature of the virtual reality environment was recognized early on, and an environmental extension was developed to enhance collaboration possibilities among users. So the environment characterized by its containing tools might be the context of this scenario. In continuing research along these lines, it might be interesting to consider in which form the information simulated in 3D can be transferred in order to be presented on different devices, or how different kinds of hardware might be recognized and used by a third user in an auxiliary environment.

In short, in relation to this thesis, we understand the context as any information, which can enhance the forthcoming aspects of the system. Knowledge of context improves users’ work, interaction and communication with others. In all our case studies, the designers’ effort to visualize this context information including work documents, submission events, time and volume was remarkable but also essential to the success of their work as a whole.
Parallel to the concept of intelligent space, in which space is highly interactive and active [Alb04], context information can add reacting capability to applications considering the contextual change of the user. To promote the forthcoming database with regard to context-awareness, the forthcoming database concept could be adapted using the following components:

- Context information repository: The forthcoming database offers the possibility to define awareness information as data objects appropriately provided to the application. This might be easily used to define context information and used as such. Thus, a context information object might be comparable to a widget in the concept of Context Toolkit. On the other hand, a kind of repository such as the space repository of the Gaia system might be necessary to allow the forthcoming database to function as a context-aware system for several different applications. This repository could reduce multiple definitions of the context and increase the economical use of the context information.

- Context information interpreter: A context information interpreter might be necessary to promote the ability of the forthcoming database to connect diverse applications environments such as 3D VR, internet and PDA applications. The information shown using large data sets in 3D VR might be made presentable in a small display or by reducing the dimensions. Vice versa, the interpreter could be used to explode information obtained through small display devices into 3D immersive scenes.

- Context information coordinator: Assuming that multiple applications will share the context information in the integrated system environment, the coordination of the context information should be guaranteed by a coordinator. This component should help the process of sharing by managing the access, changing, blocking, and releasing of information.

Integrating the context-aware concept into the forthcoming database requires system extensions at the architectural level and for handling multiple applications sharing the same context information. However, it will add possibilities for enhancing the forthcoming behavior in visualization applications built using the forthcoming database.

### 6.3 Future Work

This research has shown that the database can be used differently by breaking conventional conceptions about it. There are still many aspects to be considered, which might give new perspectives to further research. The first three points of the following list are alternative concepts and the others are potential further developments of the concept presented in this study.
• **Formalization of aesthetic**

Aesthetics are often referred to in this thesis, and certainly, it is an important research topic. We can assure that the applications that served as case studies of this thesis were highly aesthetic examples and supplied the value of delighting people during use. This sometimes even offset a portion of interaction comfort that was lacking in an application and encouraged users to stay with the system. Thus, it might be interesting to formalize aesthetics in a way that their impact can be measured or calculated. Similar to interaction or usability, where research results are done quite successfully and accepted widely, it may be also a good research opportunity to define a method that can evaluate aesthetics and make them easier to apply. Since aesthetics depend so much on personal taste and intuition, it is even more desirable to work with an operable form of aesthetics.

• **Visualization information**

Many applications exist to represent data or information in a kind of visualization. These start with the data that exists already or create a description of data sets. Then, the visualization concept is developed. Sometimes, the visualization concept and data sets are developed separately without having any relation to each other and brought together when advantages or needs are pointed out. But works done the other way around are rare. For example, a library of the 19th century with many old and modern books inside is such a beautiful environment for people. The visualization of the library may be made in a 3D virtual space or by using pictures. However, we do not hear about works automatically creating a kind of data structure out of the picture or visualization. As already mentioned, we experienced how people are capable of many different ways of thinking. This might be a visualization concept for people thinking with tables or object diagrams. So, a reduction process to transform metaphors contained in a library to collections of entities might be another opportunity for thinking about art.

• **Automated data transformation**

In research conducted in this area until now, we have data sets on one side and, on the other side, a visualization metaphor concept. In between, all data transformation procedures are executed. Let us say then the process is a pipeline: data sets - transformation - metaphor. A general method, which is so general that it does not care what data sets or what visualization metaphor is at the end of the pipeline, might be implemented. This would be a method where we could bring any data sets or metaphors at both ends and create data transformations accordingly. Thus, the work of the middleware would be greatly reduced. Moreover, convention breaking visualization concepts benefit from much more freedom in defining values in the application.

• **Preparation of data through analysis of the user interaction**

The preparation of certain data sets makes a database behavior forthcoming,
especially for the performance of an application with large sets of data in the visualization. In our system, preparation was handled depending on the application designer’s concept. As the application might be for multiusers and some kind of collaboration could happen with respect to a large amount of data, it might be interesting to prepare data sets with regard to user interaction with the data. This may require a quick modeling of user interaction and interest. Also, the visualization environment might allow much more freedom in interaction both conceptually and through a wide range of tools, which is not the case of a general application. Furthermore, the user interaction should not be pre-defined or guided by the application.

• **Connection to the OODB**

  This thesis only considered relational database systems. We are convinced that the problematic handled in this work will be conceptually the same for the object-oriented database. However, it might be interesting to investigate the possible differences compared to relational database systems. It might be also possible to include the database interface, if it exists, and to connect the data modeling components layer in the forthcoming database framework.

• **Collaboration data.** It was unfortunately never allowed to work with the data sets that were used in shared mode and modified at the same time for collaboration. A collaboration environment in which users interact with the same data at the same time would improve the system and also require some modifications. It is imaginable that the immersive environment of two people inside of a cave results in a very different kind of collaboration. It then contains two aspects of collaboration, both face-to-face collaboration and distance collaboration, in one application.

• **Extension of communication layer**

  At the time the prototype was developed, the Java version was not able to integrate much other multi-media equipment. Now it is possible and also imaginable to add further connection components into AwareDB. These include PDA connections, cellular phone connections, etc.

• **Multi-visualization with same data in dimensions**

  During the third case study, it was recognized that the system limitations, such as specific equipment needed for immersive environments, very much exclude the potential circle of users. Thus, in our opinion, an improvement of 3D immersive system environments would be to develop an application representing the same data sets in a 2D environment in which the data sets are streamed from one to another and users interact in 2D and 3D immersive worlds. In doing so, the transformation work and data transfer will be intensive. To reduce the effort, some kind of meta data could be used to describe what is seen in one environment or what kind of interactions are happening. And the real data access might happen through the separate connections established from
both environments to the database. This work would require a kind of bridge of two different concepts of dimensions and could improve the usability of the 3D immersive environment.
List of Figures

1.1 Information visualization, groupware and database technologies are three backgrounds which the forthcoming database concept grew out of. The key ideas, including 3C (communication, collaboration and cooperation), transparency, aesthetic representation, and initiative behavior, helped us formulate the forthcoming database concept. 2

3.1 Interactions happening during application development among visualization, database system, application designer and users. The application designer is the most active subject in the interaction. 29

3.2 The database services act as active elements in interaction with functionalities simplifying and facilitating the application designers’ and the users’ work. 29

3.3 Architecture of forthcoming database framework containing three component layers: data modeling components, cache components and communication components 33

3.4 Communication between User-User, Database-User, Database-Domain and Domain-Domain Handler 37

3.5 Example of the process using the framework 42

3.6 Collaboration Model 43

4.1 Graphical interface for making tables visible or invisible. It shows tables read out from a course application database. Especially, the attribute *thedate* of the table *works* is processed as visible on the figure. 50

4.2 Graphical interface for setting application designers interest using the visible tables of the database. The interface shows that the application designer wants that all users get informed, in case a update transaction with respect to the value of the attribute *exercise* of the table *works* is evoked. 52

4.3 Graphical interface for setting user interest. Compared to the interface above (figure 4.2), it shows the limited possibility of a user in setting their interest, either by saying *inform me* or not. 53
4.4 Comparison of settings among users and application designer. It shows a cutout of an evaluation site. The comparison related to two data tables, computers and vacations, are shown. The yellow color is chosen to express the table, which is updated last. In the table in yellow, the data type vacations, possible transactions, and interest settings designated by the application designer are shown. The last column shows how many users are interested in getting the information the designer defines as system behavior.

4.5 On the left, the users' connection to the framework system AwareDB, which is a part of the figure 3.4 of chapter 3, is highlighted in red. On the right, the detail view of the highlighted part is shown. For each user of the application, a socket to the system is opened and kept open for a session. The socket thread handler on the framework side is responsible for managing the synchronized communication arriving through the sockets. It also controls the information distribution to the appropriate client and communications among users.

4.6 On the left side, the framework and its Java class ClientApplet are shown. The visualization applications need to create a class that extends this ClientApplet class to dock to the framework, as shown on the right side. Consequently, the control of the communications is taken over by the framework.

5.1 A model of the attributes of system acceptability, from J. Nielsen [Nie93], page 25

5.2 W1: Overview of the teams and all works submitted. On the left column, three teams are shown and on the right side of each group, works done by each group are shown.

5.3 W2: an interface showing all works of a specific team. This interface shows works submitted by Johannes and Russel over the 5 lecture weeks.

5.4 W3: shows detail information about a submitted work including concept description and resulting images.

5.5 W4: all recent work submissions are shown in the event window.

5.6 W5: a Java application window distributing new events in the database.

5.7 ER (Entity-Relationship) schema diagram of the course database.

5.8 Course participants described their use of the course environment.

5.9 Overview of how participants searched for information on other teams' work and solutions to their problems.

5.10 Main window of Virtual Office showing users who are reachable and the place where users are present. It also allows users to exchange instant messages.

5.11 Shows the entrance time of a user.
5.12 Holiday Planner: a part functionality of the Virtual Office allowing people to declare their vacation time and to find out that of their Coworkers. .................................................. 99

5.13 Developer’s tool to manage data. It especially allows the developer to choose which data attributes are made visible to the user. From that moment on, the user can define their interest related to the actions being executed on the attribute. .................................. 100

5.14 Developer’s setting for system behavior: to define how the system has to react with regard to different kinds of data transactions. A similar interface is provided to each user. ......................... 101

5.15 Evaluation showing users’ and the application designer’s perspectives on the application data. .................................................. 102

5.16 “The Living Web” in the CAVE environment, by Lopez-Gulliver [LGSM02], page 12 ........................................... 109

5.17 Cubes Delegate: different view during navigation in a document database, see in ICCS project’s homepage [Too], Delegates → Cubes Delegate .................................................. 109

5.18 Basic visualization method of the system VisDB. The resulting data items of a query are sorted according to their relevance with respect to the query and the relevance factors are mapped to colors. Finally, they are arranged on the screen from the middle. Each window shows different dimensions. See in [KK96] ....................... 110

5.19 Infoticle scenes: (a) A user immersed in an infoticle system (b) Data sources of students and money are spreading out in the scene (c) Density view of infoticles ......................... 111

5.20 A module handling specific domains for the data transport to different kinds of application environments is added to the framework. In this case study, a C++ specific module was implemented. .................. 115

5.21 A process placed in the middleware framework gets activated by the slave process, in this case data acquisitions process, to prepare data before the data requesting query arrives at the data cache. ........ 119
Bibliography


[Ban91] L. J. Bannon. *Design at work: Cooperative Design of Computer Sys-

tems*, chapter From Human Factors to Human Actors, The Role of

Psychology and Human-Computer Interaction Studies in Systems De-


[BSS+01] D. Bartz, D. Staneker, W. Straßer, B. Cripe, T. Gaskins, K. Orton,


[BST+00] R. Bosch, C. Stolte, D. Tang, J. Gerth, M. Rosenblum, and P. Han-


[Bür99] M. Bürger. *Awareness bei der Gruppenarbeit mit gemeinsamen Ar-

beitsbereichen*. Hubert htz Verlag, Wissenschaft, 1999.


[CFB00] S. Ceri, F. Fraternali, and A. Bongio. Web modeling language (webml): a modeling language for designing web sites. In *WWW9 Confer-


B. J. Davis. An interview with richard saul wurman, our role as technologist. *design matters, Society for technical communication*, 3(2), December 1998.


Sun Microsystems Inc. Setting the class path. Website. [http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/classpath.html](http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/classpath.html), Sun Java Software.


