A Quest for Integrated Object Oriented Approach in Software Design

Zenon Chaczko[1], Witold Jacak [2], Ryszard Klempous[3], Jan Nikodem[3],

[1] Faculty of Engineering, University of Technology, Sydney, Australia, e-mail: zenon.chaczko@uts.edu.au
[2] Witold Jacak, FHS, Hagenberg, Austria, e-mail: witold.jacak@fh-hagenberg.at
[3] *Wroclaw University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wroclaw, Poland,*
e-mail: *{ryszard.klempous, jan.nikodem}@pwr.wroc.pl,*

Abstract *Ever growing interest in multimedia, web applications and embedded systems has created completely new markets and related software applications. New software solutions need to be scalable, configurable and require handling a massive amount of transactions. For this purpose Object-Oriented technology offers significant benefits as no longer 'traditional' relational database is seen to be effective in handling massive sets of images, maps, videos, sounds, financial instruments, engineering diagrams or molecular structures. The paper offers a unique insight into practice-based training in design of management systems with high performance capabilities of commercial OODBMS in Software Systems Design.*

INTRODUCTION

Surprisingly, despite the high usability and availability of database technology most of the software around the world still consists of old fashioned file systems and legacy data. Although, the RDBMS technology have been around for many years and the fact it was found to be ideally suited for scalar type of data such as: id numbers, names, address fields, phone numbers, amounts, volumes and other similar data, it represents only a minority of traditional business-processing data systems that potentially could utilise the relational database technology [17, 18]. It is expected that in the coming few years, a bulk of legacy software will be rebuilt. These applications will be redesigned and rebuilt for a variety of reasons. Often, evolving business requirements, new concepts and up-to-date technologies (e.g., web capabilities, mobile/PDA access, modern GUIs, global mapping/GIS and others) need to be incorporated. For example, some application may require including spatial/temporal concepts into a RDBMS. In most cases, this requires a considerable re-structuring of the relational database and often violates the data independence principle. As a result database designers need to know a great deal more about the application domain theme in order to determine what type of multi-dimensional entity is required and what type of valid operations are permissible. Also, performance of RDBMSs suffers greatly as a result of significant modifications required to incorporate spatial/temporal information and the fact majority of RDBMSs are based on a loose-coupling approach. Choice of the technology platform often represents a key consideration when migrating or rebuilding software systems. One of the most critical of all choices to be made is the selection of Database platform. Only few years ago the most obvious platform choice for decision makers, was the relational database, - a DBMS solution and the real question became one, which related to choice of vendor. These days however, those who have to deal with DBMS related decisions (i.e. software system architects and DB analysts) face several alternatives to relational database that have to be seriously considered. A selection of a database platform carries a degree of uncertainty as the database technology is in a state of considerable turmoil.

It was not until 1995, when one of the major players on the database market, the IBM, provided its customers with first object-relational extensions to its prime product - DB2. At the time however, an addition for object orientation support merely reflected occasional user requirements for software systems to handle storage and access for multimedia data types such as images, charts, graphs, maps, video and audio clips. Ever growing interest in Internet, multimedia and global services [5] has created completely new markets and related software applications (i.e. e-Commerce, e-Government, e-Education, e-Health, etc.). These new software applications often use Object-Oriented technology, need to be scalable and require handling a very large number of transactions per minute. Many of internet-based applications require their database servers to efficiently manage large numbers of user-defined structured data. In such applications, no longer 'normal' relational database could handle large number of images, maps, video, sound, financial instruments, documents, engineering diagrams or molecular structures. At the same time, even embedded software solutions become more sophisticated and needing to handle a large number of application-domain-specific types. The ODBMS technology is well suited to meet these requirements.

Many established academic institutions are conscious of the fact that training in objects, object-orientation and software development platforms that support OO methodology is important for various commercial and engineering applications [16, 19]. In the past, many of the educational institutions have opted for a safe path and introducing selected aspects of object-orientated design and neglecting the fact the object persistency requires addressing the issues of object-orientation in all software layers and components including the Database. Here we also argue that an intermediate approach of training students in using hybrid, post-relational

DB systems can be as effective, smoothing the transition to full OODBMSs without compromising theoretical approaches to OO data models and OO design. Additionally, such approach can be seen as justifiable considering how sensitive these issues still are to a large number of software development firms and, users of software applications.

RELATIONAL DATABASE MANAGEMENT SYSTEM

Since early 1980s, a large number of software systems relied on Relational Database Management Systems (RDBMS) in fact RDBMS are still the preferred choice for many developers. In 1970, Edgar F. Codd published a seminal paper titled - "A Relational Model of Data for Large Shared Data Banks". The article outlined a general theory of data arrangement, which was accepted as the definitive model for RDBMS. Earlier hierarchical and network models were used in practice but the idea of using tables was a major breakthrough. In Codd's relational model themes are represented through relations (tables), as shown in Figure 1. In the relational database model a table is simply a collection of zero or more rows. From the *Set Theory*, a relation is defined as a set of tuples over the same scheme; hence a row of information is also called a *tuple* of a relation that is constructed in a database table while each column is called an attribute. As always we dealing with sets, the order in which rows are listed does not change the meaning of the data in them. Attributes have alphanumeric types (e.g. character, string, time, date, real, logical). Recent additions are BLOBS types. Relationships are not explicit, but implied by values in specific fields (e.g. foreign keys). Each row in a table (tuple in a relation) must be distinct—that is, no two rows can have exactly the same values for every one of their attributes. Therefore, there must be some set of attributes (it might be the set of all attributes) in each relation whose values, taken together, guarantee uniqueness of each row. Any set of attributes that can do this is called a super key (SK). Super keys are a property of the relation (table), filled in with any reasonable set of real-world data. The database designers select one of the super key attribute sets to serve as a unique identifier for each row of the table or primary key (PK) of the relation. Codd continued to develop his concepts and in 1974 he defined 12 rules of the relational model for database management. In collaboration with Chris Date the model was extended and documented in so called normalised forms for DB design.

Wine Table

Column Name	Data Type	Length
[key]	int	4
Name	varchar	50
Brand	varchar	50
Country	varchar	50
Length	tinyint	1
Vintage	tinyint	1
[Australian Price]	money	8
[US price]	money	8
Availability	binary	50
[Tasting Notes]	varchar	8000

FIGURE 1: A "WINE" TABLE IN A RELATIONAL DB MODEL

One of such the normalised forms, called - the Boyce-Codd Normal Form, is named after Codd. Ironically, IBM was not an early adopter of its relational database technology, as it was afraid it would have adverse effect on sales of their mature IMS database products that used navigational models. IBM's commercial rivals quickly adopted Codd's ideas and to remain competitive IBM finally had to join in. Codd's model provided a blueprint for development of IBM's database system called "System R". The system used a dedicated Structured English Query Language (*"SEQUEL"*), (later renamed as SQL) to manipulate and retrieve stored data. Consecutively, SQL was adopted as a standard by the ANSI in 1986 and by ISO in 1987. Over 80s, the relational DBs gained a wide acceptance, although many vendors had initially only added a relational veneer to their older technology. Relational databases do a fine job in mapping data into a set of tables of rows and columns; Hence from the beginning they gained a universal acceptance. Since their initial arrival more than three decades ago, the RDBMS have significantly matured. This is reflected by variety of tools that have been developed and made available for RDBMS application. There are many software solutions being built even today that are heavily depend on them. However, since the middle of 1980s, thanks to the introduction of Object Oriented technology and C++ language, a new concept of connecting objects with databases is gradually receiving limited acceptance and this is happening despite relational databases' dominance.

OODBM SYSTEM

Object-Oriented concepts were first developed in the beginning of 1980s, further enhanced when Object-Oriented programming languages such as Smalltalk or C++ first emerged. In the past, for many projects it was not unusual that newly developed software became obsolete already at the time of its introduction. The Object-Oriented software development approach was found to be an effective, if not the only realistic - solution to the problem. Object-Oriented software systems are modeled around concept of abstract software objects that define the characteristics (names, properties and the behaviour) of entities in the real world. Behind publicly known interface, software objects hide (encapsulate) the internal complexity of their data structures and behaviors. This special ability of objects enables them to be frequently used in construction of a vast range of software components without needing to know the specific details of their internal implementation.

An object-oriented database management system (*ODBMS, OODBMS, ODB, OODB,*), often called *"the object-oriented database system"* is the result of the integration of OO programming language capabilities with database capabilities. An ODBMS makes database objects appear as programming language objects in one or more existing programming languages. Object database management systems extend the object programming language with transparently persistent data, concurrency control, data recovery, associative queries, object navigation and other capabilities [1].

In early years of 1980s, a natural reaction was to think that within few years, all Object-Oriented software systems soon would also use Object-Oriented databases. The idea was that since we deal with objects only, we should somehow try to store the whole objects in a persistent data store as is, instead of mapping them into tables. Additionally, Object-Oriented approach promised to enhance development of databases by such useful aspects as modeling power, extensibility of systems, code reuse and ease of program maintenance. It was expected that within short period of time, all Object-Oriented based software systems would also use OO databases. Only now we can see that these assumptions were, in fact, way off the mark. As far as the popularity of Object-Oriented software is concerned there is yet no correspondence with the popularity of OODBMS. Object-Oriented systems, based on the principles of user-defined data types, along with inheritance and polymorphism are firmly established in the software industry; yet, it is only recently OO programming has itself really taken off by the storm. This is happening mostly due to ease of programming in new generation of OO languages (i.e. Java, C# and many others) as well as availability of efficient methods, techniques and tools for Object-Oriented software development (i.e. Design Patterns, Frameworks, Middleware, IDEs, RADs). Researchers and developers started to learn early that there exists a large class of applications for which relational database systems and other more conventionally structured database systems are too limited [1], however for most of applications developers were quite happy to get things easier and simpler in context of relational databases [18].

A CASE STUDY: ADVANCED REAL ESTATE SYSTEM

Work on software development projects, in large teams and following realistic scenarios, is at the core of the SSD subject's teaching and learning [2, 5, 16]. Main goals are to synthesise students' knowledge, find skills gaps in, enrich their learning process, enhance communication skills and provide a significant project experience. In Spring semester of 2005 a medium size software group of SSD (15) students were implementing the Advanced Real Estate Agency Management (aDREAM) system. The objective for the development group was to develop an Advanced Real Estate Agency Management system taking an integrated Object Oriented approach. The system was to provide a range of efficient business services and interfaces. The system was to be equipped with web-based user interfaces that enable various classes of users to easily interact, query and access a variety of information about property types and assets attached to each of these properties (i.e. apartment communities, family homes, commercial properties, etc.).

System Boundary Model

As a result of the software system requirements analysis an initial System Boundary Model (SBM) as shown in Figure2 was proposed. The initial SBM provided an abstract view of

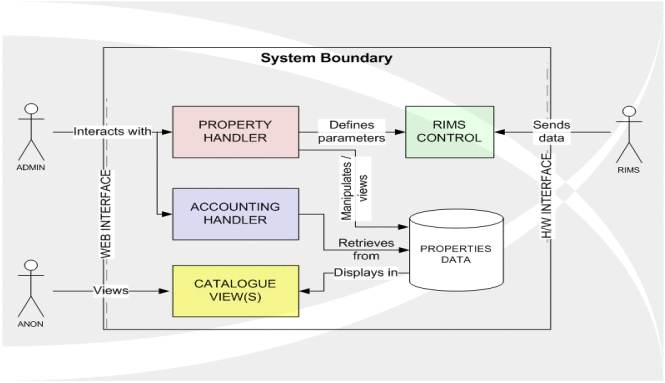

FIGURE 2: ADREAM SYSTEM BOUNDARY MODEL (SBM) [12]

the system as a whole, thus allowing the system vision to be communicated to various stakeholders. The SBM is a useful tool for conveying the perceived solution at the top level to non-technical customers and a initial blueprint to derive the system architecture and High Level Design. The aDREAM system was refined down to a set of interacting entities, so that the new system remains adaptable to changes. These entities are purely conceptual, and thus do not restrict the actual design of the system. Each of functional requirements in the system's SRS document has been elicited within the SBM context. Each functional, non-functional, and behavioural requirement elicited from the User Requirements Document is refined and specified according to the boundaries posed by the SBM. Each requirement shall have a direct relationship to an SBM component. The design drivers of the system were scalability (through the plug-and-play components) and integrity of all data residing in, and extrapolated from the system. The system was required to support two main groups of users through the Web Interface. The *Anon* type of unregistered users with explicit access to the system/site and *Admin* users such as tenants, vendors and agents. A*dmin* and *Anon* users have access to the catalogue views (CV) but only *Admin* type of users have access to both the Accounting and Property Handlers components; The main components of the aDREAM system are the following subsystems:

- **Accounting Handler (AH):** This component specifically deals with all necessary accounting based transactions. Worth noting is the fact that entry to the Accounting Handler component relies on the assigned Admin user privileges. The information, data retrieved, modified and stored by this component, is collated in the Properties Data repository.
- **Catalogue View (CV):** This component presents the graphical display of information in read-only format. Access to this component is not restricted to a specific User type.
- **Property Data Repository/Reporting**: Centralised data repository that accepts queries from Property Handler, Accounting Handler, CV components. This component can be viewed as a subsystem of each component, i.e. all components of the SBM have a defined reporting sub-component attached.

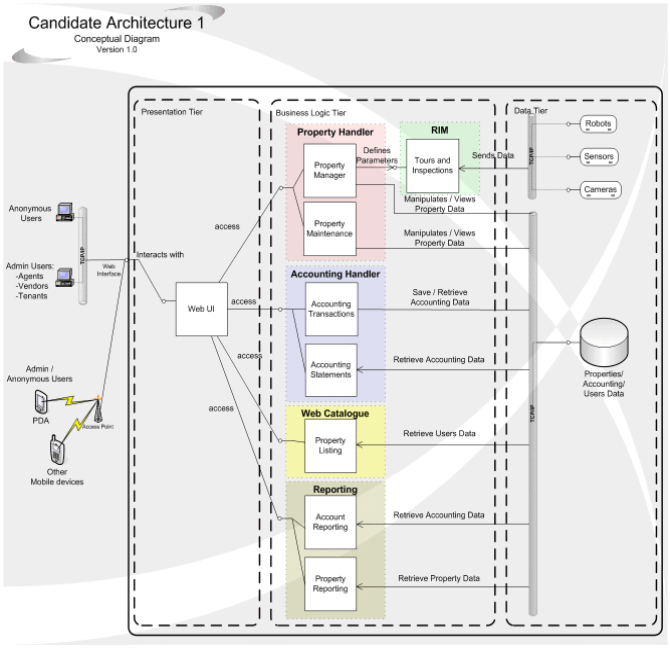

FIGURE 3: ADREAM SYSTEM ARCHITECTURE [12]

System Architecture and Design

- **Property Handler (PH):** This component is specifically for transactions concerning property maintenance, property details, problem tracking and assets. Access to the Property Handler component depends on the assigned user privileges.
- **RIMS (Remote Inspection Management System).** The most important element of RIMS was the remote-controlled camera/robot). Hardware components related to the RIMS component include various video and image sensors, image sensors.
- **RIMS Control (RC):** Parameter and reporting templates as defined by the Admin user. The Admin user can define this parameter and reporting templates.

The aDREAM SBM provided a blueprint for discussion realisable architectures derived by decomposing each functional component into smaller parts, analysing the relationships between these parts and clasifying architectural components into distinctive layers. In the preferred architecture the Reporting component was integrated into the Property Handler whilst in the rejected architecture the Accounting subsystem was incorporated into the Property Handler. The business logic components of the selected architecture are connected directly to the database whereas in the rejected architecture the Data Access Tier shielded the database and the Web Catalogue was dependent on RIM component. In the selected solution the Catalogue component does not need to depend on both the data access tier and the RIM component. In the chosen architecture the data access tier controls access to all the property and accounting objects so the Business Logic components can access the DB directly. The Web Catalogue component allows streaming pre-recorded virtual tours to users by accessing data directly from the database thus delivery of video and access to property/accounting objects could execute much faster.

Software architects were concerned with such quality attributes as: security, usability and reliability. In order to satisfy the security concerns of system stakeholders, the selected system architecture is expected to allow for implementation of the following:

a) *Restricted system access:* System's anonymous/ unauthenticated users are prevented access from areas of sensitive information and administrative access rights. Authentication over HTTP would allow authorised users access to system's areas that they have been given access to.

b) *Restricted database access:* Due to the tier structure of the architecture, compromise of the web server still prevents unauthorised access from the database.

System Concurrency Issues

One of the most important design issues in the aDream system was problem with concurrency. When deployed the system is sharing of common resources between deployment systems running in parallel. This required employing techniques for coordinating execution, data exchanging and scheduling processing to optimise efficient use of resources. The concurrency diagram below depicts this conceptual aspect of the architecture.

System Reliability. In order to satisfy the reliability concerns of system stakeholders: The following strategies were sought to be can be achieved with the proposed architecture:

a) *Modularity*: Thanks to OO design of the system (i.e. components/connectors), the system can continue to function even if a component becomes unresponsive or crashes. This could relate to maintenance tasks on components, as it might not be recommended to offline the entire system for performing scheduled (unscheduled) maintenance. Statistical information into component usage can determine when during the day/week/month maintenance on a particular component can be performed with minimal impact to users.

b) *Multi-system Deployment*: Deployment of the system over more than one physical system/machine, downtime due to hardware issues/faults should not offline every component. This should enhance security, as well as robustness (due to additional resources).

c) *Automatic Fault Recovery*: In the case the web server (which runs the web interface) or the DB suffers an error, these subsystems will attempt to automatically recover from the fault, either by rejecting the request that caused the fault, or even perform a soft restart/reboot (if the error was critical). In either case, the users, administrators, and other related staff could be informed to remedy the issue as soon as possible.

d) *Database Backup/Data Mirroring:* Loss of important data can be prevented with regular backup of the database.

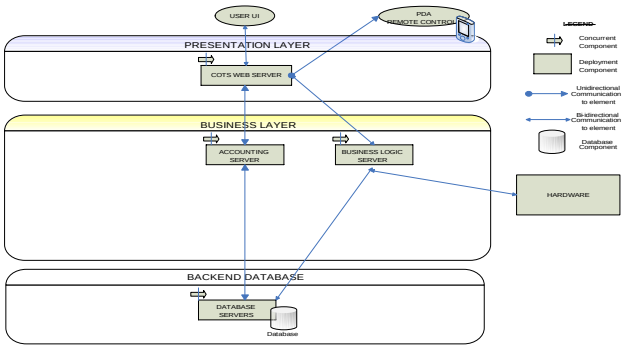

FIGURE 4 : ADREAM CONCURRENCY DIAGRAM [12]

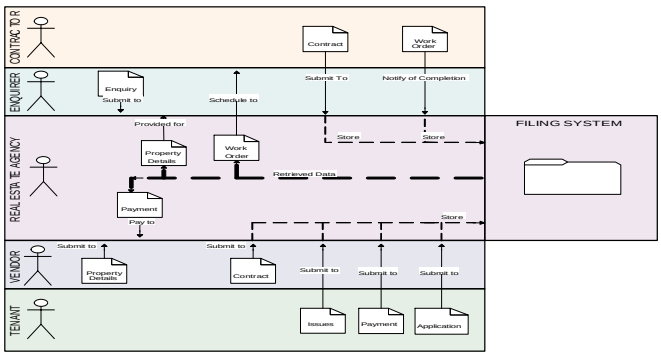

FIGURE 5 : ADREAM BUSINESS MODEL [12]

In case the main database is corrupted, or unrecoverable from error, the backup database can be used until the problem is solved with the main database.

System Usability is reflected by extend the system satisfies concerns of system stakeholders the selected architecture should allow such strategies as: profiling registered users' with search and viewing history, so that information of possible use/relevance to the user can be derived and if the wish displayed to them from their usage history, it can be displayed to them. An easy-to-use web interface need to be designed instead of a GUI-based application, the ease, simplicity and general familiarity of a website structure can be employed to make a users use of the system satisfying/without frustration. By focusing on developing a simple-to-understand navigation for the website, an overall design that does not strive too hard in being flashy, and applying common sense, one can avoid the mistakes many websites have. The system's architecture should allow variety of access methods. Access to the system cannot be restricted to the PC; a user can still access all parts of the system via a web-enabled PDA or mobile phone. Integrated OO Design Approach

In the aDREAM design process the following main steps were executed:

1. Definition of the aDREAM system Business Model to capture the key business processes of a real estate agency.
2. DB Design – An Object-Orientated DB was developed following requirements from the business model.
3. Detail OO System Design – Development of low level classes diagrams to breakdown previous high level components to low level classes.Site Hierarchy –

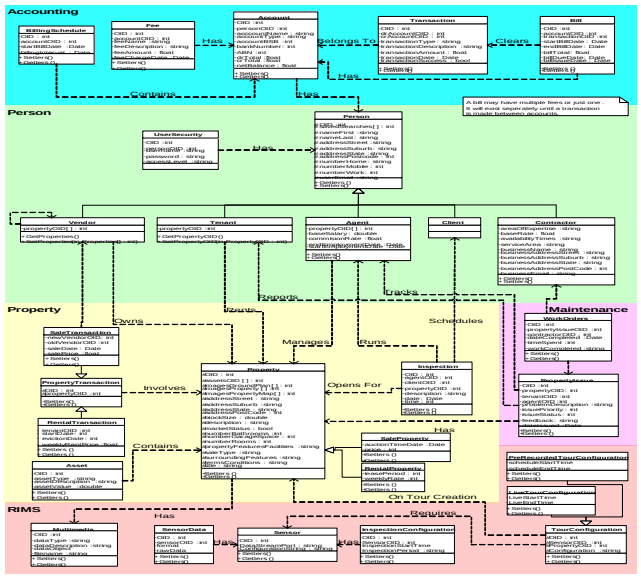

FIGURE 6: DB CLASS DIAGRAM [12]

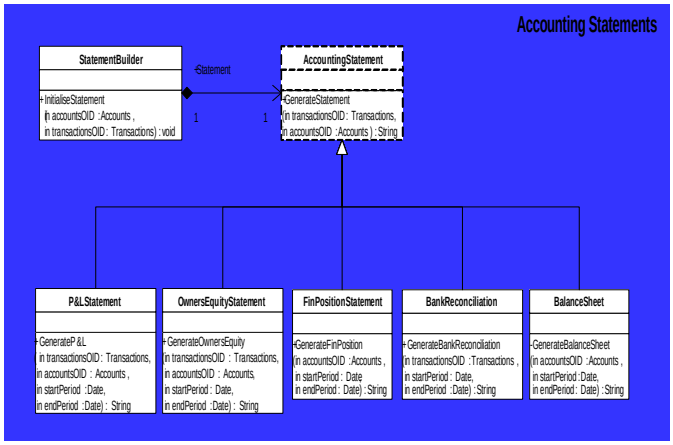

FIGURE 7: THE BUILDER PATTERN IN ACCOUTING STATEMENTS OF ADREAM SYSTEM [12]

4. Development of web-oriented GUIs.
5. Use Case Analysis – definition of cases scenarios, which detail web pages mappings to components in the business logic tier.
6. Technology Assessment – Justification of various technological choices to be considered in the project while preserving a holistic OO Design approach.

The aDREAM Business Model represents the operations and data exchanged between the actors in of real estate agency management system. The process used to define the business model involved understanding how the system might operate without the use of IT technology. This approach facilitated understanding of the basic operations of the business and highlighted the various dependencies and processes involved. The primary purpose of business model analysis was to isolate those areas that would be replaced by software. The agent is the central actor in the model who is responsible for much of the operations. Agents have to interact with other actors as well as to collect, retrieve, disseminate and store various data. When a real estate agent is required to deal with many actors

regarding many different properties, a bottleneck may occur at the real estate agent. By replacing manual processes of the real estate with software processes, the real estate agency becomes much more efficient and manageable. Many operations can be automated using software and improve level of efficiency offered by the real estate agent to their customers. For example this may include having a web catalogue so a real estate agent might not have to search property data for customers wishing to browse through properties they may be interested in. The filing system is probably the main aspect of the business that can be replaced with a software system (see Figure 5). The storing of data as documents has its obvious deficiencies limiting scalability, performance, maintainability, and integrity of the data stored. In terms of storage, there is a limit to how much data can be physically store in say a filing cabinet.

The retrieval and updating of the data also becomes more efficient as the use of queries is much more efficient than physically searching for documents. Other benefits of using software to aid the real estate agency management could include the ability to have virtual tours and inspections for prospective clients. This promotes availability and serves as a form of marketing. Apart from clarifying the possible bottlenecks of the system and the areas that could possibly be enhanced with software, the business model offers an insight into the types of data that is processed in the normal operations. Superficially, the main sources of data in the system would be the data concerning properties and the supplementary data related to these properties. Additional, supplementary data may include property issues, tenant details, vendor details and work orders. These data would be fundamental to operations of the agency and this is where the qualities of data maintainability and integrity that the software offers become most valuable.

Design of Object-Orientated Database

An OO Database was applied to meet the user requirements of the aDREAM system. As one of core functional requirements, the aDREAM system was required to store virtual tours of properties so that potential customers can view the property over the web. The aDREAM was also required to store traditional persistent data relating to transactions, property descriptions and property reports (maintenance, financial, etc). Due to the varying data requirements, the aDREAM system needed to manage complex data (pre-recorded virtual tours / inspections, still images, etc) as well as traditional / simple data types such as text (property descriptions, reports, etc), numbers (transactions, fees, etc), and dates. To reduce complexity associated with implementation, as well as improve performance both simple and complex data needed to be integrated. Taking into account these issues, the decision had to be made if to select an OODB that would best fulfil the initial aDREAM data requirements or mitigate risks associated with unknown technology and resort to a choice of traditional relational databases (i.e. mySQL, INGRESS, SQL Server, etc.). The Cache post-relational database from Intersystems was selected to implement the Database for the aDREAM system. The choice to use Cache has been justified as follows:

1. OO methodology was used for the design of aDREAM system. DB model design has also been created using OO concepts as shown by DB class diagram in Figure 6. Since Cache provides an object database, and considering that the aDREAM DB design also uses OO designs, there is a direct mapping between the database design and rest of the implementation. This undoubtedly saved time and resources during the project's implementation phases.
2. Similarly, due to the nature of OO methodology, objects in any OODB can not only store atomic types, but also other type of objects. This results in object oriented out-performing relational databases as there was no need for complex queries and multiple joins.
3. Post relational Object Caché DB[2] well fit the criteria and initial requirements placed upon a database for aDREAM. Cache allowed storage of complex data types as well as integration of complex data with simple data.
4. Within the UTS/Intersystems Campus program Caché Object DB was readily available for SDD team in the UTS development laboratories.

aDream System Detail OO Design

From initial system architecture a simplified class diagram of the high level design of system was developed. Further decomposition of detailed OO Design for a Dream was enhanced through application of various OO design patterns to various trackers of the aDream system subcomponents. Initial OO design of system components, their implementation, maintenance and use was simplified. For example implementing the Simple Factory Patterns allowed for design simplicity where each tracker is essentially providing the same functionality, albeit with different data classes. Also, it was possible to create an abstract base tracker class that captures all the common functionality in one place, aiding maintenance and reducing duplicated code.

Design Patterns in Accounting Statements

In Accounting Statements component consideration has been given to application of a creational design pattern as it deals with the creation of instances of objects. Statements are to be instantiated by the Accounting Statement component and accordingly provide for the adaptation of the creational design patterns. For variety of reasons some creational design patterns were not suitable for use component others like the *Builder* pattern (see Figure 7) were found easily adaptable. Builder pattern works to build an object constructively whilst maintaining independence from other builders. The builder design pattern allow in the Accounting Statement for independency of each builder from others builders while working to improve the modularity of the reporting function

[2]Caché is a post-relational database from InterSystems Corporation

for additional future statement functions, also it allows variation in the internal representation of the statements generated. As the *Builder* constructs the final statement progressively, there is greater control over each final 'statement' that a builder constructs. Te Accounting Statement class was decomposed into new elements. *The Factory* design pattern has not been applied to the Accounting Statements component as the Accounting Statements class considers which statements to instantiate and creates. This feature of the Accounting Statements class was found to be not concordant with the factory design pattern. The Accounting Statements class should be instantiated when called by the Presentation layer. As a result the class must be public which does not respond to the characteristics of the factory design pattern. Consideration was given to the application of the abstract factory design pattern to the Accounting Statements class. Again, it has not been adopted for as the dependency on the Accounting Statement object to return related classes of objects is not desired for this component since there is not direct relationship between the main Accounting Statement class and various statement functions.

OBSERVATIONS

There are similarities between .Net and J2EE platforms that made them initially both suitable as a platform of choice for the development of aDREAM system using the integrated OO approach. Both platforms offer an environment for design of business logic, on either the client or server side, and both contain APIs specifications for data access, directory services, and remote applications. The selection criteria for suitability of a given framework were based using the following criteria:

Interoperability and Customisation

.Net developed applications can share information with applications running on other platforms through Web Services (WS). Although J2EE has begun to develop this feature, it was found not on par with the .Net. WS played an important role in the aDREAM project with a strong reliance on developing a web-based solution for users to access the system both locally and remotely. .NET's strengths lies in its strong support of WS development, which was the focus of the project, and it lend support to the decision to go with the .NET framework. .NET's event-driven programming model made programming the web easier than in J2EE. MS tools offer rich GUIs that deliver content to web browsers. J2EE tools are considered a bit more difficult to use than .Net applications but J2EE does allow for a higher level of customization to business rules, and is more suitable for mission-critical applications. .Net may not offer the same level of extensive customization that J2EE provides, but it was found adequate for the project, and the difficulty in developing using J2EE tools removes that as an option for the project, despite the customisability that it provides, as the SSD was developing a simple system within limited constraints.

Availability and Usability of Tools

When developing in Java, the developers have a plethora of tools available to them, whereas MS Visual Studio .NET provides a single IDE for building .NET applications. Visual Studio .NET has many features and the developer is able to build applications without actually leaving VS .NET. This strength of .NET reduces the overall difficulty in development. From the key differences that affect the choice between the .Net and J2EE [19], it can be seen that the .Net framework was found more suitable for the aDREAM application due to its low-cost, rapid turnaround times as well as interoperability with WS and Cache DB technologies

Criteria for choice of middleware

As with the assessment of the other technologies, architectural qualities were a driving force behind the decision to use WS. Relative to .Net Remoting, Web Services conforms to the W3C standard promoting portability. As Web Services use a standard protocol, the presentation layer can be deployed using non-.Net technologies such as Java. .NET remoting requires both the client and the server to be built using .NET, limiting the user to a Windows platform [19]. WS can only be hosted on Internet Information Services (IIS) by default. Hence, IIS security facilities such as secure socket transmission; authentication/authorisation services are inherited by Web Services. .Net Remoting can be deployed using binary communication channels to improve performance. Developers, however, might need to implement their own authentication/authorisation and secure socket transmission methods (i.e. SSL). WS are highly robust because they are always hosted on IIS. Within IIS, it can take advantage of "fault isolation" mechanisms such as "thread safe" and "auto restarting". For real-time, high performance, systems the .Net Remoting in combination with high performance OODB can be more applicable. Instead of sending remote objects over binary channels, WS encode/decode objects using the SOAP protocol. Due to the overhead related to the packaging and removal of such packaging, WS are often much slower than its binary format. With .Net Remoting, states can be kept between each client request through client-activated or singleton objects.

Web Application vs Windows Application

There are features of both web applications and windows applications that offered certain benefits for using them in the aDREAM. For example, both are compiled rather than interpreted, making them faster than the other options, and both provide the capability to develop a suitable application for aDREAM system. However, there were differences between the two, which influenced the choice of using web applications over windows applications. aDREAM system was envisioned to be used by hundreds or thousands of users, and this supported the choice of web apps approach. The choice of .NET framework simplified the development of web applications. As the .NET framework was used for system development, this naturally determined the choice of the web applications over the windows applications. Using .NET for

development of web based applications offered time savings in testing, integration, installation, and maintenance. Also, the database management and data storage for hundreds or thousands of user transactions would be quite expensive and arduous without thorough application of OO methodology and highly scalable Cache DB technology.

Additionally, updating the software to many users can be considerably more difficult using window based applications and they may require a single developer to change the code running on the server to affect all the users of the system whereas the updating of windows applications would require users to download and install the updates. WS applications run inside a browser and developers can ignore addressing screen resolution, keyboard/mouse interaction, and other standard issues when developing Windows Apps. This provides a significant benefit, due to the project's limited timeframe. Additionally, web applications are more portable and adaptive [19] than a windows application due to the fact that all a client needs is a web browser. Windows applications require the user to have the .Net framework installed on their system. Web applications were found more suitable for the distributed aDream system environment than Windows apps, providing cost savings and ease of development. Also, the scale of the system and the integrated OO design that included all layers of the application justified support to web apps.

CONCLUSION

During SSD exit interviews (oral exams) students confirmed that initial fears of unfamiliar approaches and apprehension toward including OODB in the system design were totally unjustified. In fact very positive experiences resulted from applying an integrated approach (integration of OO data model with high performance post-relational DB). Students were truly amazed and surprised how seamless and effective was the integration of the presentation, business and persistent objects layers when using an OO approach through all stages of software design and encompassing all layers. Most students commented positively on their experiences, highlighting how fond they were of the final results of the project and their newly acquired skills in using both proper OO design methodology and applying high performance industrial tools (DB Cache). The subject's main objective - to cater for thorough academic training of students and equipping them with professional skills that are in high demand in the ICT industry, has been achieved.

ACKNOWLEDGMENT

I would like to acknowledge the work carried out in Spring 2005 by aDREAM team members: Chiu C., Dao X.H.,Gango, N.,Gosal R.,Huynh P.,Lam J.,Le M.H. Luong P., Palocz D.K., Pandey N., Ta V.H., Tang L., Tariq R., Ton J. and Wong W. Special thanks go to Dr Mike Silvester from Intersystems for his tremendous support and help in our teaching program.

REFERENCES

[1] Zhang W., Ritter N., *The Real Benefits of Object-Relational DB-Technology for Object-Oriented Software Development*, Advances in Databases: 18th British National Conference on Databases, LNCS, Springer Berlin / Heidelberg Vol. 2097 , UK 2001.

[2] Chaczko Z., D. Davis, V. Mahadevan, *"New Perspectives on Teaching and Learning Software Systems Development in Large Groups"*. In *5th Internat Conf on IT Based Higher Ed and Training ITHET '04* p278

[3] Chaczko Z., URS for Smart Real Estate System, SSD Spring 2005, UTS, Australia.

[4] Chaczko, Z, Davis J. D, Scott C., *New Perspectives on Teaching and Learning Software Systems Development in Large Groups-Telecollaboration"*, IADIS International Conference WWW/Internet'04 Madrid, Spain, October 2004

[5] Doerry E., Klempous R., Nikodem Jan, Paetzold W.: *"Virtual Student Exchange: lessons learned in virtual International Teaming in Interdisciplinary Design Education"*.: 5th International Conference on Information Technology Based Higher Education and Training: ITHET 2004. Proceedings, Istanbul, May 31 - June 2, 2004.: IEEE

[6] Dhawan, P. 2005, Performance Comparison: .NET Remoting vs. ASP.NET Web Services http://msdn.microsoft.com/library, accessed 20/9/2005

[7] Gaffaney, D. , Klima, C. J2EE vs. .Net AIIM E – Doc Magazine Silver Spring: Sep/Oct 2003 Vol. 17 Issue 5

[8] Groh, M., 2003 'Are Windows Apps Dead?', *Access Advisor*, viewed 19/9/2005, <http://doc.advisor.com/doc/09317>.

[9] Johansson, M., 2002, 'Debate - .NET vs PHP: Top 6 Reasons to use .NET', *ASP & .NET Tutorials*, viewed 19/9/2005, <http://www.sitepoint.com/article/v-php-top-6-reasons-use-net>

[10] Patton, T, Choosing between Web services and remoting, http://builder.com.com/5100-6371_14-5841906-2.html,accessed 8/9/2005.

[11] Sarrel, M. D. The Big Decision: J2EE or .Net PC Magazine. New York: Sep 2, 2003 Vol. 22 Issue 15.

[12] SSD Documentation for the Smart Real Estate System, S2005, UTS, Australia.

[13] Sliwa, C. .Net vs. Java Computer World Canada Jun 14, 2002 Vol 18 Issue 12

[14] Thangarathinam, T., .Net Remoting versus Web services, http://www.developer.com/net/net/article.php/11087_2201701_3, viewed on 9/9/2005

[15] Utley, C., '.NET Web services vs. remoting: Which one will work best for your app?', *Web Services*, 2003, viewed 19/9/2005, <http://builder.com.com/5100-6389-5034970.html>

[16] Chaczko Z. Dobler H., Jacak W., Klempous R., Maciejewski H., Nikodem J., Nikodem M., Rozenblit J., Paz Suarez Araujo C., Sliwinski P. *"Assessment of the Quality of Teaching and Learning Based on Data Driven Evaluation Methods"*, ITHET'6, Sydney, Australia, 2006.

[17] Celko, J. *Data and Databases: Concepts in Practice*, Morgan Kaufmann Publishers Inc.; 1st ed., June 19, 2006.

[18] Smith K. and Zdonik S., *Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems*, OOPSLA '87 Proceedings, 1987.

[19] Klempous R., Nikodem J., Nikodem M.: *"Our experience in using test systems based on three different technological platforms"*.: 5th International Conference on Information Technology Based Higher Education and Training: ITHET 2004. Istanbul, May 31 - June 2, 2004.