**UNIVERSITY OF HERTFORDSHIRE**

**Faculty of Engineering and Information Sciences**

**Modular MSc Honours in Computer Science (Software Engineering)**

**7WCM0031 Software Engineering MSc Project (Online)**

**Final Project Report**

**January 2016**

***Development of a distributed data and document management system for ‘MSc Properties’***

**Mr D L Edwards**

**SRN: 07156987**

**Supervised by: Thiago Matos**

Abstract

Acknowledgements

Table of Contents

**No table of contents entries found.**

1. Introduction
   1. Introduction to the project

For my MSc Computer Science Dissertation (Software Engineering), I decided to solve the problem of data and document management through the implementation of a distributed system for a fictional estate agent called ‘MSc Properties’.

The reason for me undertaking this project is that there is an ever increasing demand for private rented accommodation due to changes in the social housing market, social benefits offered, and unemployment rates, resulting in less social housing being available and reduced benefits for people on a low income, who would normally need to make use of social housing and/or benefits offered by the government, meaning families are unable to afford the private rents of their current area they live in and need to move to a different area they can afford to privately rent.

* + 1. **Report Structure**

During this report I will document the aim and objectives of this project and explain the problem background in more detail, looking at the different factors that contribute to the problem identified.

I will then document the research and literature review I carried out in order for me to identify the different software engineering techniques and technologies available to solve the problem.

I will then go on to document the design of the distributed system, outlining the different software engineering techniques chosen, and why these were chosen. I will then explain how these were used to enable me to structure and manage the project, and just as importantly allow me to produce diagrams that virtualized the structure, behaviour and interaction of the distributed system.

Furthermore, I will document the implementation and testing of the distributed system, explaining which techniques and technologies I chose, and why they were the best solution for this project, given the available resources. I will then go on to provide an analysis of the test results to draw some conclusions on the validity of the software produced.

Lastly I will document my evaluation of the project as a whole, outlining what went well, what didn’t go so well, and what I would do different if I was to do the project again. I will then go on to stating whether or not I have successfully achieved the project aim and objectives.

* 1. Project Aim

The aim of this project is to tackle the issues of data and document sharing across the Internet by developing a distributed data and document management system for a fictional estate agent called ‘MSc Properties’.

The distributed system should allow ‘MSc Properties’ to share business data and documents across the Internet, whilst providing data security and integrity. ‘MSc Properties’ requires the distributed system to be maintainable, dependable and usable, which means I will explore the different techniques that support program specification, design, validation and evolution of software.

* 1. Project Objectives
     1. Core Objectives

Analyse ‘MSc Properties’ current business processes by week 5.

Complete literature searches and review of existing data management systems, identifying the software engineering models, methodologies, tools and metrics used in the development process by week 12.

Set out functional and non-functional requirements for the development within the requirements document by week 9.

Ensure required resources are available for the entire project by week 9.

Carry out risk assessment by week 10.

Set out the distribution mechanism I am going to employ for the distributed system by week 13.

Develop a suitable data management system model that meets the requirements defined by week 15.

Write test scripts to test the implementation of the system outlined in the development model by week 15.

Develop a suitable database to handle the business data and import dummy data into the database by week 16.

Develop a suitable application to handle the business processes and connect to the database to store the business data by week 24.

Develop a suitable search facility so users can search for information stored in the database, and should be implemented by week 24.

Develop reporting functionality so certain users can report on business performance indicators by week 24.

Develop a log in facility for users, allowing for restricted access, and to prevent unauthorised access and should be implemented by week 26.

Test the system using the test scripts created, ensuring the test results are above the acceptable failure rate defined in the requirements by week 33.

Develop and test a user manual by week 31.

Evaluate the project in a report to detail the entire development and outline what went well and what could have been done better by week 34.

* + 1. Advanced Objectives

Develop a website to advertise services offered to potential customers/suppliers. Customers/Suppliers will be able to register and submit a service request through the website and should be implemented by week 26.

Develop document management facility that allows for documents to be stored electronically, and should be implemented by week 26.

Develop a home screen which provides a live feed of the tenancies and leases due to expire by week 26.

Develop a reset password facility, so users are able to reset their password if they have forgotten it allowing users to establish access to the system. This should be implemented by week 26.

* 1. Project Background

‘MSc Properties’ is a fictional estate agent with a number of sites nationwide across England. Due to the current unemployment rates, and recent legislation changes resulting in local councils being able to house homeless families outside of the local borough [25] and benefit caps [23] meaning families have to move out of their local borough due to not being able to afford local rents [25], ‘MSc Properties’ require the need to be able to transfer customers between sites, meaning the transfer of data and documents across sites that could be 100’s of miles apart.

‘MSc Properties’ require me to develop a distributed system to create and manage their property portfolio, as well as creating and managing customer/supplier/employee accounts. They require this so that data can be stored on a server or locally and all the officers of different ‘MSC Properties’ sites will be able to access this data. The system will have a login facility to provide restricted access for users, and will also allow managers of ‘MSc Properties’ stores to manage their employee accounts. The system will also allow ‘MSc Properties’ managers to report on business data.

‘MSc Properties’ have allocated an office manager to the project to assist with analysis of current business processes, map system requirements, and communicate back to ‘MSc Properties’ with the work that is occurring in project meetings, and relaying any project document back to the business for sign off. From the project meetings the below business analysis and system requirements were compiled.

* + 1. **Business Analysis**

‘MSc Properties’ currently have an electronic filing system, where customers, landlords of properties and potential employees fill out an application form to request a service from ‘MSc Properties’. Anyone applying for a service will have to provide their personal information. Customers will also need to provide the properties they have an interest in letting. Landlords will have to provide the property information for the property they would like managed. Potential employees will have to provide the job role they are applying for.

Part of the landlord/customer/employee sign up process is to bring in a number of documents confirming their information, and once customers have agreed on a property to let, once ‘MSc Properties’ have agreed to manage a property for a landlord, or once ‘MSc Properties’ have agreed to take on an employee, contracts will be drawn up and signed by both by an ‘MSc Properties’ employee and the client, and is called the agreement.

Each agreement will result in a transaction account being created and documented in an Excel spreadsheet, which will record all transaction information for the different agreements, for example, rent payments from tenants, salary payments to employees, or lease payments to landlords. All of this information captured in the form along with any documents and spreadsheets will be stored electronically on the computer of the officer managing the case, except for the employee information which is stored on the office manager’s computer.

‘MSc Properties’ currently advertise their property portfolio and job role vacancies through pictures in their store windows. Office managers carry out a revenue monitoring exercise each month, going through each transaction account, and carrying out budget monitoring and cash flow forecasting, along with reports for the month’s business activity. Also there is a staff hierarchy within ‘MSc Properties’ where specific tasks and responsibilities are assigned to certain staff roles meaning that not all staff can carry out each task due to privileges.

‘MSc Properties’ will also require a number of different system environments to enable them to carry out training to employees outside of the live data system. Also as ‘MSc Properties’ require the system to be maintainable, dependable and usable, I also believe an additional environment needs to be produced to enable testing to occur if a system upgrade was to take place in the future. This means ‘MSc Properties’ will require three system environments, which are a live, clone and test environment.

* + 1. **System Requirements**

NEED TO DO SYSTEM REQUIREMENTS

* 1. Literature Review

As previously outlined, I will now document the literature reviews I undertook to allow me to understand the different options available to me to enable me to successfully achieve the project aim and objectives.

* + 1. **Design Methodologies**

The problem I am trying to solve by undertaking this project, requires a piece of software to be developed for me to successfully achieve the project objectives surrounding data and document sharing. But as well as trying to solve the business problem surrounding data and document sharing, there is also the problem of producing software that is maintainable, dependable and usable, which will require me to undertake literature reviews into different design decisions I will need to make when designing the system.

* + - 1. **Software Development Approach**

The first design principle I am going to discuss is the software development approach. This is because as this is a large project, it requires the project to go through a structured development process to give the project the highest possibility of successfully achieving the project aim and objectives.

The piece of literature I am going to review for the software development approach is the article called *“Software Quality & Agile Methods”* written by *M. Huo, J. Verner, L Zhu and M.A. Babar.* This article looks at the quality of software produced when comparing the Waterfall Model and Agile methods, and specifically how agile methods can achieve high quality software even if the process is not linear and a complete requirements specification has not be developed prior to the design and implementation stage of the development.

The article then goes on to conclude that “agile methods do have practices that have Quality Assurance abilities, some of them are inside the development phase and some others can be separated out as supporting practices. The frequency with which these agile Quality Assurance practices occur is higher than in a waterfall development and lastly, agile Quality Assurance practices are available in very early process stages due to the agile process characteristics”. The below diagrams show the different methods and Quality Assurance techniques undertaken within the Waterfall Model and Agile Methods.

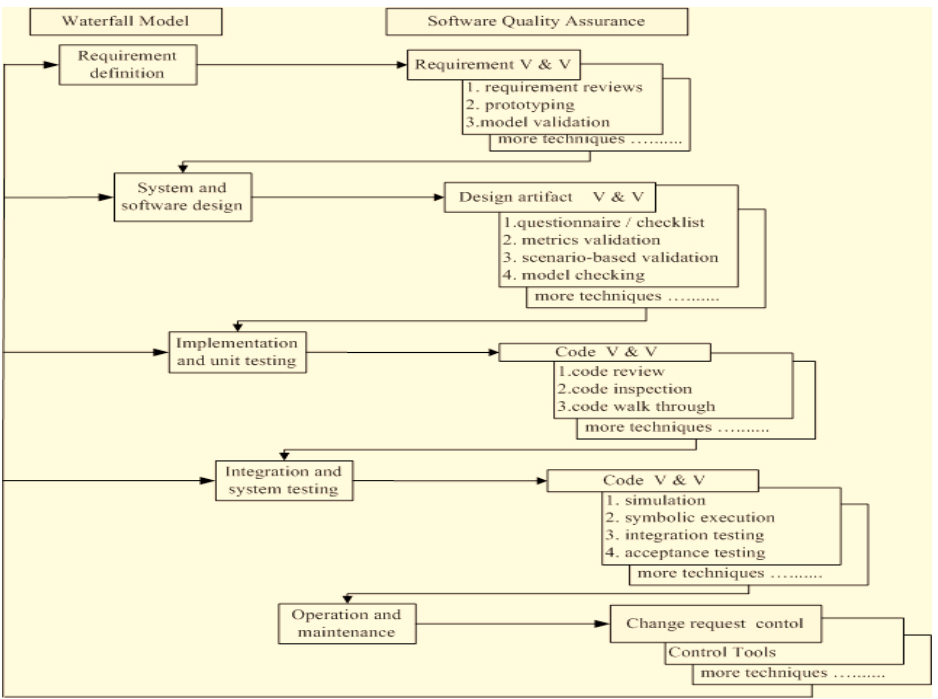


Fig 1 – Waterfall Process Model with Quality Assurance Techniques

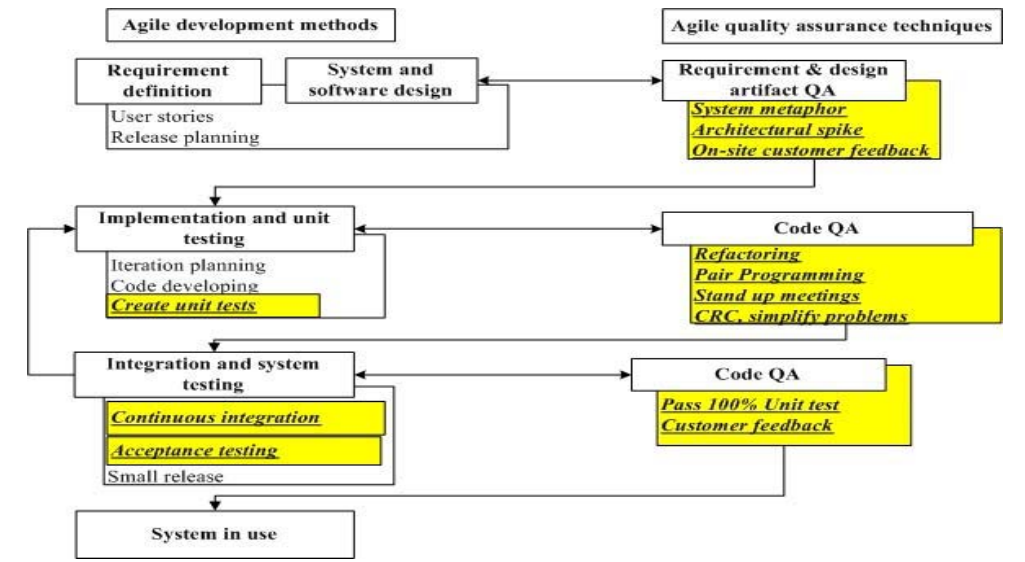


Fig 2 – Agile Development Methods with Quality Assurance Techniques

* + - 1. **Modelling System Behaviour**

The next design principle I am going to discuss is modelling system behaviour. This is because this project centres on software development, and part of the aim is to produce software that is maintainable, dependable and usable, so it will require me to produce high quality software, and by modelling system behaviour and understanding how the elements of the system will interact with each other.

The piece of literature I am going to review for the modelling of system behaviour is the article called *“Designing Concurrent, Distributed, and Real-Time Applications with UML”* written by *H. Gomaa.* This article looks at two areas, the software design method called Concurrent Object Modelling and Architectural Design Method (COMET), which is an example of a Unified Modelling Language (UML) based method, and the different modelling required for concurrent, distributed and real time applications using UML.

The article explains that “In the requirements model, the system is considered as a black box and the use case is developed… In the analysis model, the emphasis is on understanding the problem, hence the emphasis is on identifying the problem domain objects and the information passed between them… In the design model, the solution domain is considered, so the analysis model is mapped to a concurrent model”. These different models are what I will have to consider when going through the software development process for this project.

Below is a list of different techniques to model system behaviour which I have come across during the research for this project:

* Use Case Diagrams – A representation of a user’s interaction with the system, showing the relationship between the user and the use cases they are involved in [Wiki].
* Data Flow Diagrams – A graphical representation of the “flow” of data through an information system [Wiki].
* Class Diagram - A static structure diagram that describes the structure of a system showing the system classes, their attributes, methods, and the relationships amongst objects [Wiki].
* Entity Relationship Diagram – A data model for describing the data or information aspects of business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as relational database [Wiki].
* Class Responsibility Collaborator (CRC) Models – A brainstorming tool used in the design of object-oriented software, documenting the dynamics of object interaction and collaboration [Wiki].
* Sequence Diagrams – An interaction diagram that shows how processes operate with one another and in what order, showing object interaction arranged in time sequence [Wiki].
* Storyboards – A graphic organizer in the form of illustrations or images displayed in sequence for the purpose of pre-visualising a motion picture, animation, motion graphic or interactive media sequence [Wiki].
  + - 1. **Design Patterns**

The last design principle I am going to discuss is design patterns, and most importantly the design patterns I can employ within the software I am going to produce. Again as with modelling system behaviour previously, this project centres on software development, and part of the aim is to produce software that is maintainable, dependable and usable, so it will require me to produce high quality software, and by implementing design patterns, it will allow me to produce software with high cohesion, low coupling, encapsulation, and other metrics of software development which indicate high quality software.

Ian Sommerville explains design patterns as a description of accumulated wisdom and experience, a well-tried solution to a common problem, and the Hillside Group puts it as “Patterns and Pattern Languages are ways to describe best practices, good designs, and capture experience in a way that it is possible for others to reuse this experience” [].

Below are some design patterns I have come across during my research for this project:

* Observer pattern – A software design pattern in which an object, called the subject, maintains a list of dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods [Wiki].
* Singleton pattern – A software design pattern that restricts the instantiation of a class to one object [Wiki].
* Strategy pattern – A software design pattern that enables an algorithm’s behaviour to be selected at runtime [Wiki].
* Creational pattern – A software design pattern that deals with object creation mechanisms, trying to create objects in a manner suitable to the situation [Wiki].
* Iterator pattern – A software design pattern in which an iterator is used to traverse a container and access the container’s elements [Wiki].
* Composite pattern – A partitioning software design pattern, which describes that a group of objects is to be treated in the same way as a single instance of an object, allowing clients to treat individual objects and compositions uniformly.
* Inheritance – Is when an object or class is based on another object or class, using the implementation to maintain the same behaviour, and is a mechanism for code reuse and in programming languages that support inheritance, produce an “is a” relationship between sub classes and its parent class. [Wiki].
* Object Composition – Is a way to combine simple objects or data types into more complex ones, and are a critical building block of many data structures. Composition can be regarded as a relationship between types: an object of a composite type “has an” object of a simpler type [Wiki].
* Object Relational Mapping – Is a programming technique for converting data between incompatible type systems in object-oriented programming languages, and in effect creates a “virtual object database” that can be used within the programming language.
  + 1. **Development Methodologies**

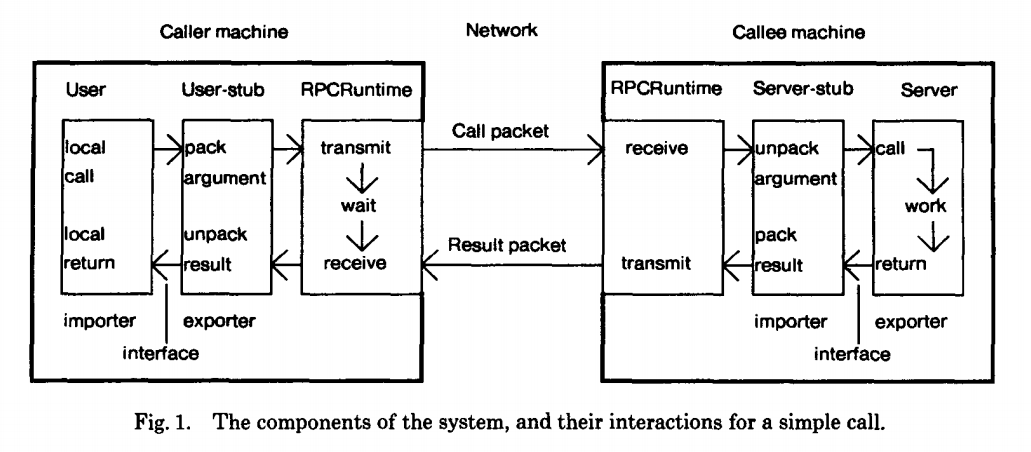
I am now going to discuss the different development methodologies I uncovered during the course of the project, whilst undertaking the literature review.

* + - 1. **Networking**

One of the major problems I am trying to solve by undertaking this project, is to be able to produce a system that allows ‘MSc Properties’ to share data and documents across the Internet, whilst ensuring that the validity of the data and documents being shared are maintained. For me to do this, it has meant that I have had to undertake a literature review into the different networking technologies that will enable me to successfully achieve the project aim and objectives related to networking.

The first piece of literature I am going to review for Networking is the article called *“Implementing Remote Procedure Calls”* written by A.D. Birrell and B.J Nelson. This article talks about the options that face the designer implementing remote procedural call (RPC) functionality and the considerations that need to be made when making decisions on this type of system.

The article states “when making a remote call, five pieces of program are involved: the *user*, the *user-stub*, the RPC communications package (the RPCRuntime), the *server-stub*, and the *server*… When the user wishes to make a remote call, it actually makes a perfectly normal local call, which invokes a corresponding procedure in the user-stub. The user-stub is responsible for placing a specification of the target procedure and the arguments into one or more packets and asking the RPCRuntime to transmit these reliably to the callee machine. On receipt of these packets, the RPCRuntime in the callee machine passes them to the server-stub. The server-stub unpacks them and again makes a perfectly normal local call, which invokes the appropriate procedure in the server. Meanwhile, the calling process in the caller machine issues pended awaiting a result packet. When calling the server completes, it returns to the server stub and the results are passed back to the suspended process in the caller machine. There they are unpacked and the user-stub returns them to the user.” This process is represented in the below figure.



The Remote Procedural Call functionality highlights the type of technology I could implement to successfully achieve the project aim and objectives of data sharing amongst ‘MSc Properties’ hosts that are at different locations.

The second piece of literature I am going to review for Networking is the article called “*Push vs. Pull in Web-based Network Management”* written by *J.P. Martin-Flatin*. This article talks about two models of network management, which are “The Pull Model” and “The Push Model”, which represent two well-known approaches to exchanging data between two hosts with a distance between them, which is one of the major problems I am trying to solve during this project.

The article states “The pull model is based on the request/response paradigm, the client sends a request to the server, then the server answers, either synchronously or asynchronously. This is functionally equivalent to the client “pulling” the data off the server. In this approach, the data transfer is always initiated by the client, i.e. the manager. The push model, conversely, is based on the publish/subscribe/distribute paradigm. In this model, agents first advertise what Management Information Bases they support, and what Simple Network Management Protocol notifications they can generate; the administrator then subscribes the manager (the Network Management Station) to the data he/she is interested in, specifies how often the manager should receive this data, and disconnects. Later on, each agent individually takes the initiative to “push” data to the manager, either on a regular basis via a scheduler (e.g., for network monitoring) or asynchronously.” The article then goes on to state that “the pull model, well suited to ad hoc management, and the push model, well adapted to regular management”.

* + - 1. **Document Management**

The next development area I am going to discuss is document management, which is one of the biggest areas of ‘MSc Properties’, as the business deals with numerous documents that need to be stored, and available for access by any host at different locations.

The piece of literature I am going to review for Document Management is the article called *“Electronic Document Management: Challenges and Opportunities for Information Systems”* written by *R.H. Sprague, Jr.* This article talks about a number of benefits gained from implementing a document management system, such as improving the publication process, supporting organisational processes and communication amongst people and groups, improving access to external information, creating and maintaining documents, maintaining corporate records and lastly promoting training education.

The article states “Documents are stored electronically, shipped over a network and printed when and where they are needed, resulting in reduction in obsolescence, elimination of warehouse costs, and reduction or elimination of delivery time.”, the article later goes on to explain “The benefits of Electronic Document Management for these applications are, quicker access to the documents, more efficiency in the search process, simultaneous access by several people to the most current version of the document, and reduced cost of printing and distributing documents”.

Below are some document management frameworks I have come across during my research for this project:

* Apache JackRabbit – Is a content repository which implements the Content Repository for Java (JCR), with support for structured and unstructured content full text search, versioning, transactions, observation and more [].
* Modeshape – Is a distributed, hierarchial, transactional, and consistent data store with support for queries, full-text search, events, versioning, refrences, and flexible and dynamic schemas, which implements the Contant Repository for Java (JCR) [].
  + - 1. **Task Scheduling**

The next development area I am going to discuss is task scheduling, and in particular task scheduling in real time systems, this is because the distributed system that will be developed for ‘MSc Properties’ during this project will be a real time system, dealing with the processing of scheduled tasks.

The piece of literature I am going to review is the article called *“Application of Real-Time Monitoring to Scheduling Tasks with Random Execution Times”* written by D. Haban and K.G. Shin. This article talks about the calculation of execution time for posteriori tasks (calculation of execution time requires experience with the given task) scheduled within real time systems, and that the worst-case execution time is usually used to ensure that enough time has been allocated for the task to be completed, and discusses the drawbacks with this sort of approach and alternatives to this method.

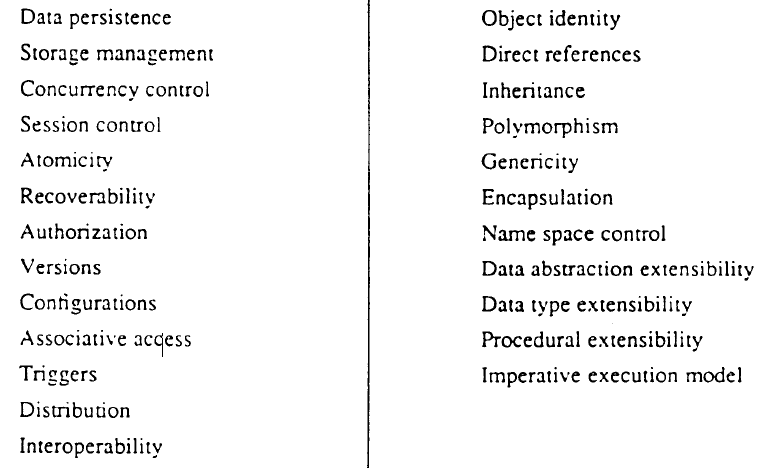
The article states “real-time tasks are usually scheduled based on their worst-case execution time, and since the worst-case execution time can be several orders of magnitude larger than the true execution time, scheduling tasks based on the worst-case execution time can lead to severe underutilization of CPU cycles and/or incorrect decision on the schedulability of tasks i.e., some tasks are declared to be un-schedulable even if they can be completed in time”.

Below are some task scheduling frameworks I have come across during my research for this project:

* Quartz project -
* Haban & Shin project
  + - 1. **Database Management Systems**

The next development area I am going to discuss is database management systems, and in particular data models based on object-orientated concepts. This is because I have decided to develop a distributed system using object-orientated concepts, such as objects, inheritance, Encapsulation etc., which means I will need to adopt a database management model based on these object-orientated concepts.

The piece of literature I am going to review is the article called *“An Introduction to Object-Oriented Database and Database Systems”* written by *M.L. Horowitz.* This article looks at the possibilities of combining most of the desirable features of database systems with desirable features of the object orientated model of computation and the below diagram outlines these features.



The article later goes on to explain the issues that can arise when combining the database and object-oriented model concepts, by stating that “First integration should occur without impedance mismatch. In particular, language support for object-oriented database services should be orthogonal and transparent. Second, integration should not lose any advantages of existing data models. For instance, object-oriented programming does not support data independence inherently, so features such as relationship support and query joins should be provided. Finally, integration presents an opportunity for introducing new desirable features”.

Below are some database management systems I have come across during my research for this project:

* Microsoft Access – Is a Database Management System (DBMS) from Microsoft that combines the relational Microsoft Jet Database Engine with graphical user interface and software development tools [Wiki].
* MySQL – Is an open-source relational database management system (RDBMS) and the most widely used open-source client-server model RDBMS [Wiki].
  + - 1. **Web Server**

The next development area I am going to discuss is web servers, and in particular the different web server software that can be implemented to receive and manage the Hypertext Transfer Protocol requests sent by a user of the website that will be developed to advertise ‘MSc Properties’ services to potential customers, and also manage communication between the web server and the database management system discussed previously.

The piece of literature I am going to review is the article called *“Specification and Implementation of Dynamic Web Site Benchmarks”* written by *C. Amza, A. Chanda, A.L. Cox, S. Elnikety, R. Gil, K. Rajamani and W. Zwaenepoel.* This article looks at the movement from web content being static HTML or image files, to web content becoming dynamic through the combination of a front end web server (web browser such as Internet Explorer), an application server (software such as Apache, along with server side scripting in PHP and SQL), and a back-end database (software such as Microsoft Access) and in particular identifies benchmarks for dynamic web sites by comparing 3 different dynamic web sites, looking at the bottleneck characterizations for these web sites.

This combination of technology would allow me to develop a dynamic website that can advertise ‘MSc Properties’ services to potential customers, and be updated by staff when they perform actions in the system which result in updates to the website, for example, a property being rented to a customer meaning the property is no longer available to rent and therefore should no longer be advertised on the website as available to rent.

The article explains that “We have used our implementations to carry out a bottleneck characterization of the benchmarks. Different benchmarks show different bottlenecks: the database CPU for the online bookstore, and the Web server CPU for the auction site and the bulletin board. Complex queries cause the database CPU to the bottleneck for the online bookstore. In contrast, the queries for the other applications are simpler.

Below is some web server software I have come across during my research for this project:

* Apache HTTP Server – Is the worlds most used web server software, and played a key role in the growth of the World Wide Web quickly becoming the most dominant HTTP server [].
* Nginx – Is a web server with a strong focus on high concurrency, performance, and low memory usage [].
* Cherokee – Is an open source cross-platform web server that runs on Linux, BSD, variants, Solaris, Mac OS X, and Microsoft Windows [].
  + - 1. **Graphical User Interface**

The next development area I am going to discuss is graphical user interface (GUI), and in particular the different frameworks that can be adopted with my chosen programming language to develop an interface for the ‘MSc Properties’ staff to interact with the system I am going to produce during this project.

Below are some Java GUI frameworks I have come across during my research for this project:

* Abstract Window Toolkit (AWT) – Is Java’s original platform-dependent windowing, graphics and user-interface widget toolkit, and is part of the Java Foundation Classes (JFC) [].
* Swing – Is a GUI widget toolkit for Java, was developed to provide a more sophisticated set of GUI components than the earlier AWT, and is also part of the JFC [].
  + 1. **Testing Methodologies**

As previously explained, this project will centre around developing a system to tackle the major problem of data and document management and sharing, and whilst meeting these functionality requirements of ‘MSc Properties’, the system also needs to be maintainable, dependable and usable. This means that I will need to carry out testing to ensure that the system I have developed for ‘MSc Properties’ is compliant with the specified requirements and that the system has no faults or errors at runtime of the system, or the system is at an acceptable level for ‘MSc Properties’.

* + - 1. **Unit Testing**

The first testing principle I am going to discuss is the unit testing approach to testing, and will be fundamental to ensuring each of the elements of the system work independently of each other (where elements are not coupled).

The piece of literature I am going to review for unit testing is the article called *“A Simple and Practical Approach to Unit Testing: The JML and JUnit Way”* written by *Y. Cheon and G.T. Leavens*. This article looks at ways programmers can reduce the writing of labour-intensive code for unit testing, by writing formal specifications (for example, pre and post-conditions of methods).

The article goes on to explain that “writing formal specifications instead of test code makes the programmer’s task easier, because specifications are more concise and abstract than the equivalent test code, hence more readable and maintainable. Furthermore, by using specifications in testing, specification errors are quickly discovered, so the specifications are more likely to provide useful documentation and inputs to other tools”.

Below is some unit testing framework I have come across during my research for this project:

* JUnit Testing -
* Java Modelling Language (JML) -
  + - 1. **System Testing**

The next testing principle I am going to discuss is the system testing approach to testing, and will be fundamental to ensuring each of the elements of the system work together as they should.

The piece of literature I am going to review for system testing is the article called *“A UML-Based approach to System Testing”* written by *L. Briand and Y. Labiche.* This article looks at system test cases being derived from the analysis stage documents such as use case diagrams and also looks at the functional system test methodology called Testing Object-orienTed systEms with the unified Modelling language (TOTEM).

The article goes on to explain that “Deriving test requirements from early artefacts produced at the end of the analysis development stage, namely use case diagram, use case description, interaction diagram associated with each use case (sequence or collaboration), and class diagram (composed of application domain classes and their contracts). This early use of analysis artefacts is very important as it helps devising a system test plan, size the system test task, and plan appropriate resources early in the life cycle. Once the low level design is complete, when detailed information is available regarding both application domain and solution domain classes, then test requirements can be used to derive test cases, test oracles and test drivers”.

Below are some testing technologies I have come across during my research for this project that will assist in bug tracking:

* Bugzilla
* The Bug Genie
  1. Project Plan

As this is a large project, it is very important that I planned, monitored and managed the project smoothly from start to finish. I have used a Gantt chart, which provides a graphical illustration of the schedule of the project, broken down by project objectives, with completion dates for each objective, which has helped me track the activities in the project and make changes to work being carried out if necessary. This tool has been used to manage my time and allow me to stay on schedule best as possible, as there was a lot of tasks that needed to be completed in a limited time frame. This process of project planning is outlined in a software management article [5], where the article identifies “a recent update of the Chaos Report from the Standish Group, outlines a recipe for success that includes 10 items. The first three items are executive support, user involvement, and experienced project management.”, so project management is one of the 3 key factors to successful projects.

My project Gantt chart is below:

ADD GANTT CHART

I am now going to explain how the project objectives have been successfully completed by the project deadline date of 11 Jan 2015.

1. I wrote a project document outlining the details of the project, defining project objectives, scope, risks and approaches. I can constantly refer to this document to ensure the project progresses in the correct direction.
2. I wrote a work plan outlining the project objectives, with deadlines for each objective.
3. I defined relevant resources for the project, outlining decisions made on technology, equipment and software applications to use, ensuring that I have tested equipment and software applications, and am competent with the use of the selected technologies, prior to the start of the development.
4. I kept an eye on the project plan ensuring that objectives do not overrun past their completion date (where possible).
5. I stayed vigilant and alert to early warning signs of problems occurring in the project that may have resulted in the project being delayed and not meeting project deadlines.
6. I safeguarded against my project creeping outside of scope, so as new requirements were introduced during the development process, I had to ensure these are all still within available resources and overall aim and objectives of the project.
7. I managed risks as the project progressed, and as new risks were discovered, I had to evaluate them to ensure they do not cause a major problem to the project.
8. I tried to keep my project supervisor informed of any major problems occurring during the project, and did at times, seek advice where necessary, to resolve major problems as early as possible.
   1. Relevance to target award

Software Engineering is defined by Ian Sommerville as an engineering discipline concerned with all aspects of software production (specification, development, validation and evolution), and goes on to say it is concerned with the practicalities of developing and delivering useful software [1].

My project aim is to develop a distributed data and document management system, and to do this I had to explore the different software engineering techniques and decide which are best suited to tackling the software engineering task, and then develop and implement a piece of software that successfully meets the aim and objectives of the project.

This means the work I carried out during this project fits in with my target award MSc Computer Science (Software Engineering), because I applied software engineering models I have studied during my course such as agile to my software development. I also applied software engineering methodology I have studied during my course such as Inheritance and Encapsulation to my software development. I also applied the software engineering tools I have studied during my course such as unified modelling language (UML) to my software development. Lastly applied metrics such as cohesion, coupling, bugs etc. to my software development. By me exploring and applying these different software engineering techniques it allowed me to deliver useful software to ‘MSc Properties’ which in essence is Software Engineering.

* 1. Required Resources and Skills
     1. **Hardware**
* Operating System – Windows, Solaris, Linux or OS X;
* Processor – Intel® Core™ i5-4288U CPU @ 2.60GHz (or similar);
* Memory – 8.00 GB (or similar)
  + 1. **Software**
* Platform – Windows XP or higher (or similar);
* A JDK for Java 5 or later
* An Integrated Development Environment (NetBeans or similar)
* A concurrent version system (Git or similar)
* A bug tracking and testing tool (Bugzilla or similar)
* A web server (Apache or similar)
  + 1. **Access**

I will require access to the following:

* MySQL database
  + 1. **Skills**
* Research skills
* Project management skills
* Report writing skills
* Ability to use Unified Modelling Language to model the distributed system
* Ability to write code in Java, HTML, Java Script, PHP and SQL.
* Ability to implement design patterns such as Observer
* Ability to use frameworks and API’s such as Spring and JRC respectively

I met these project resource and skill requirements, by ensuring I had the required hardware in place before development work began, I then downloaded the majority pf the required software resources and of the ones I did download, I tested these to ensure they work appropriately. Once I carried out a literature review of the required skills, methods and methodologies I could employ to meet the project aims and objectives, I then undertook exercises to ensure that I have understood these methods and methodologies before development work began and if any problems arose during the development I attempted to seek assistance from my project supervisor to overcome these issues.

* 1. Project Deliverables

NEED TO DO PROJECT DELIVERABLES

* 1. Ethics Approval

Ethics Approval is when a committee of University of Hertfordshire staff approve “any student undertaking a study involving the use of human participants which is undertaken as part of a programme of work for which the University of Hertfordshire is responsible for” [25].

My project will not require ethics approval because I did not undertake research that involved collecting data from human participants, and although my system does store business data which includes personal information, I used dummy information which replicates the personal information throughout the development.

1. Design
   1. Introduction

I am now going to discuss the design decisions I made during this project, and how I came to make these decisions.

* 1. Software Lifecycle

Considering the project aim and objectives, and also the project background, I believe an agile method is the best software process model to choose from, this is because in a fast moving business environment, software needs to be ready and available as quick as possible, and as original software requirements can quickly become out of date, it makes software developed useless very quick.

This means the software process model chosen needed to provide rapid development and delivery of software, and with the conventional plan driven software process models it can be difficult to do this because of the amount of documentation that needs to be created and signed off, and the lack of interleaving development stages makes it difficult to cope with quickly evolving requirements.

Also as there will be an office manager from ‘MSc Properties’ working on the project assisting the development, it would make sense in being able to deliver software quickly to allow for this to be evaluated and confirm the project is moving in the correct direction at each iteration of the agile lifecycle, as it can be difficult to gather exact system requirements from clients without going over the development process and having something to evaluate and add to or remove from to develop a system that successfully meets the project aim and objectives.

The agile method I have chosen is METHOD, and although I decided to use the agile method METHOD, as explained previously in the project background, a business analysis and system requirements exercise was carried out with an officer from ‘MSc Properties’, which allowed me to have a fairly strong idea of what functionality ‘MSc Properties’ required from the system to be developed.

The information gathered put me in a position where I was able to carry out a fair amount of design work for the system prior to the first development iteration, however I still used the divide and conquer technique which allowed me to break the development into smaller pieces, and accomplish one or a number of the smaller problems with each iteration of the development process. The breaking down of the development into smaller pieces enabled me to tackle each smaller task on its own and then combine the solutions to the smaller problems to provide a solution to the original problem, which meant providing a solution to the original problem was easier and more manageable.

* 1. Modelling System Behaviour

I used a number of design techniques to assist in the modelling of the system behaviour and am now going to explain what design techniques I used and why I decided to use these for the development.

However due to the size of the project and due to a level of resources available to the project, I decided that for me to successfully achieve as much of the project aim and objectives as possible, I would not be able to develop diagrams to model the entire system behaviour, and instead I selected a sub set of the system functionality to model.

* + 1. Use Case Diagrams

For this development I decided to develop a number of use cases which have been documented under Appendices A. By developing use case diagrams, this enabled me to identify the relationships between actors (roles within the system, for example a user of the ‘MSc Properties’ system, or the database which will hold all the system information) and use cases (functions within the system, for example creating a property).

The use case diagrams developed during the project were not only used to model the system behaviour, but I also used them to create test scripts for the system testing, as the use case diagrams outlined the different functions that should occur within the system, and therefore can be used to carry out the black-box system testing, which will be explained in further detail in the implementation section of this report.

* + 1. Data Flow Diagrams
    2. Class Diagram

For this development I decided to develop a class diagram which has been documented under Appendices C. By developing a class diagram, it enabled me to visualise the structure of the system I intend to develop, and allows me to document the variables and methods of each class, and how classes are related to each other, for example composition, multiplicity, inheritance, etc.

* + 1. Enhanced Entity Relationship Diagram

For this development I decided to develop an enhanced entity relationship diagram (ERD), which has been documented under Appendices D. An ERD is very similar to a class diagram, but instead of visualising the structure of the system, it visualises the structure of the database to be developed, and allows me to document the tables, the table columns and the relationships between the tables, for example one-to-one, one-to-many, many-to-one and many-to-many.

* + 1. Sequence Diagrams

For this development I decided to develop sequence diagrams, which has been documented under Appendices E.

* + 1. Storyboard

1. Implementation
   1. Introduction

I am now going to discuss the implementation decisions I made during this project to produce a distributed system from the design model produced in the previous section, and in doing so, successfully meet the project aim and objectives.

I decided the best way to go through the implementation decisions I made, was to take you through each of the iterations of the project development, as I have decided to adopt the agile method METHOD.

And although I will take you through each iteration and with each iteration I carried out unit testing and system testing, I have decided to leave the explanation of testing to its own section, where I will then break the testing into the unit testing and system testing as a whole.

* 1. Iteration Cycle 1

As this project involved developing a distributed system, for the first iteration I decided to implement the basic functionality of the system as a desktop version, which involved adopting a number of the design patterns I identified during my research for this project.

Although I implemented a desktop version of the system prior to implementing the networking functionality, I knew this was going to be a distributed system so I decided to split the desktop version of the system into two packages The two packages are a common package which would include any classes and interfaces that will be adopted by both the server package and the client package (once I have implemented the networking functionality), and the server package which will include the model and controller of the model, for the system I am to implement. The classes and interfaces produced in cycle 1 are listed below:

Server classes:

* AccountImpl – This class represents an account for ‘MSc Properties’ and implements Account from the common package.
* AddressImpl – This class represents an address for a property and implements Address from the common package.
* AddressUsageImpl – This class represents an actual usage of an address by a person and implements AddressUsage from the common package.
* AgreementImpl – This class represents an agreement for ‘MSc Properties’ between an office and a client and implements Agreement from the common package.
* ApplicationImpl – This class represents an application to ‘MSc Properties’ for private rented accommodation and implements Application from the common package.
* ContactImpl – This class represents a contact for a person or an office and implements Contact from the common package.
* ContractImpl – This class represents a contract between ‘MSc Properties’ and an employee, and extends AgreementImpl from the server package and implements Contract from the common package.
* Database – This class represents the application database that will hold all of the system data and although this cycle did not include the object relational mapping functionality, during a later cycle it will manage the connection to the MySQL database and deal with queries to the MySQL database to create, update and delete data within the database.
* ElementImpl – This class represents a system element, such as a religion or title for a person and implements Element from the common package.
* EmployeeImpl – This class represents an employee of ‘MSc Properties and implements Employee from the common package.
* EmployeeAccountImpl – This class represents an account for an employee contract set up by an ‘MSc Properties’ office and extends AccountImpl from the server package and implements EmployeeAccount from the common package.
* InvolvedPartyImpl – This class represents a household member of an application for private rented accommodation and implements InvolvedParty from the common package.
* JobRoleImpl – This class represents a job role for an employee of ‘MSc Properties’ and implements JobRole from the common package.
* JobRoleBenefitImpl – This class represents a benefit for the associated job role of ‘MSc Properties’ and implements JobRoleBenefit from the common package.
* LandlordImpl – This class represents a landlord of a property ‘MSc Properties’ manage, and implements Landlord from the common package.
* LeaseImpl – This class represents a lease between ‘MSc Properties’ and a landlord, and extends AgreementImpl from the server package and implements Lease from the common package.
* LeaseAccountImpl – This class represents an account for a landlord lease set up by an ‘MSc Properties’ office and extends AccountImpl from the server package and implements LeaseAccount from the common package.
* ModifiedByImpl – This class represents a modification to a system object, such as an update to a property, and implements ModifiedBy from the common package.
* NoteImpl – This class represents a note for a system object, and implements Note from the common package.
* OfficeImpl – This class represents an office of ‘MSc Properties’ and holds the Agreements and Accounts associated with the office, and implements Office from the common package.
* PersonImpl – This class represents a person within the ‘MSc Properties’ system, and can be associated with an employee, landlord or involved party and implements Person from the common package.
* PropertyImpl – This class represents a property that ‘MSc Properties’ did or does manage and implements Property from the common package.
* PropertyElementImpl – This class represents an element of a property, for example rent or number of bedrooms and implements PropertyElement from the common package.
* RentAccountImpl - This class represents an account for an application tenancy set up by an ‘MSc Properties’ office and extends AccountImpl from the server package and implements RentAccount from the common package.
* TenancyImpl - This class represents a tenancy between ‘MSc Properties’ and an involved party of an application, and extends AgreementImpl from the server package and implements Lease from the common package.
* TransactionImpl – This class represents a transaction for an account of ‘MSc Properties’, and implements Transaction from the common package.
* ServerImpl – This class represents the controller class of the model, and although this cycle did not include the networking functionality, during a later cycle it will act as the actual server, and deal with setting up the remote server for clients to connect to.
  + 1. **Design Patterns**

As explained previously, I decided to adopt a number of design patterns to assist in producing software that is maintainable, dependable and usable, I am now going to discuss the design patterns I adopted during cycle 1 and why these were relevant to this project.

* + - 1. **Iterator Pattern**

During this project, the iterator pattern was one of the most common design patterns adopted, this is because as I was producing a system that will hold a lot of data, and one of the functions of the system will be to search the data using different search criteria, it would be crucial to be able to traverse over lists of data to extract data that matches the search criteria.

There is a number of ways to traverse through a list, each with their own benefits and draw backs depending on the reason the programmer is traversing through the list and below I have highlighted a few examples of when I have used the iterator pattern, and why this was best in this situation.

**While Loop**

This loop will loop over a set of statements as long as the while condition is true, this type of loop is very powerful as it will only loop over the statements if the condition is true, and can be used to create an infinite loop by giving the while condition the Boolean condition true.

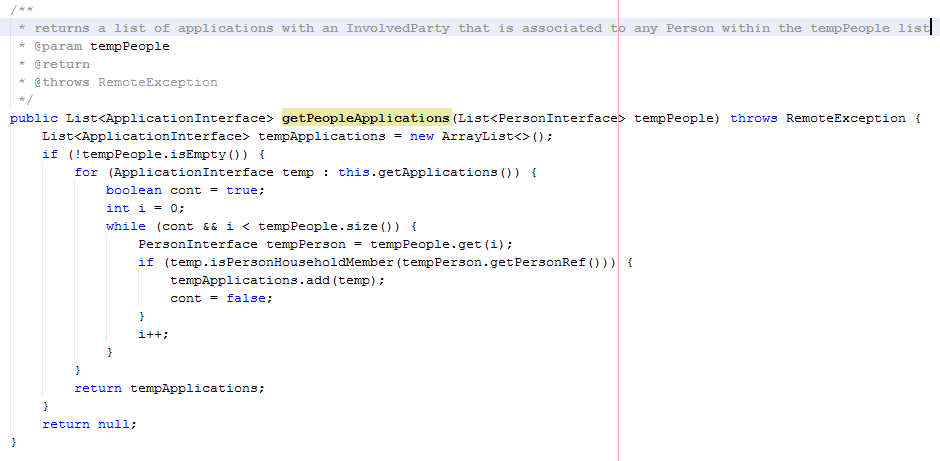


Fig x – Extract from Database, getPeopleApplications() method

This enhanced for loop from Fig x is within the Database class and as you can see I have used both the enhanced for loop to traverse over a list of all of the system applications, and again I have used this because I don’t need to know any information about the position of the element in the list, I just need the element in the list. I then decided to use the while loop because the system should only traverse over the list if the integer called i is smaller than the size of the people list, which ensures that after each iteration through the while block (where i is incremented by 1) there is still another element in the list. However, because I am checking to see if any person from the list of people is within the application as an InvolvedParty, as soon as I come across a person object that is/was a household member on the application, I don’t need to continue searching through the list of people so I also need an indicator which is always true unless I have come across a Person element which is within the current application. I used a Boolean field called cont, which is made true if a person is on the application, which on the next check of the while statement will cause the system to not go execute the while statement code and move on to the next application within the list of applications.

**Enhanced for loop**

This loop is a simpler way of doing the standard for loop and traversing a list, however is not flexible and should only be used when you need to loop over all of the elements within a list, and don’t need to know the index of the object you are retrieving [].

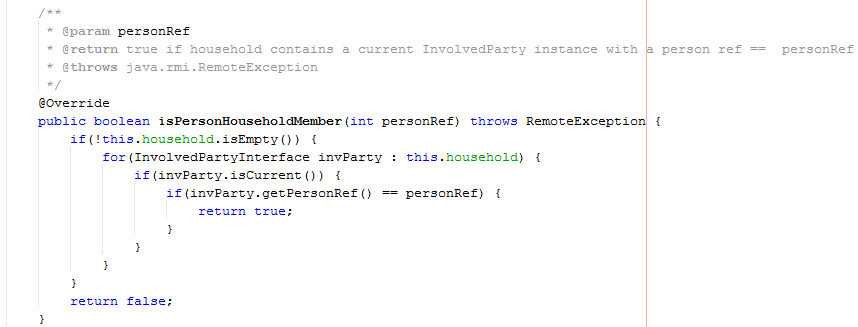


Fig x – Extract from InvolvedPartyImpl, isPersonHouseholdMember() method

This enhanced for loop from Fig x is within the InvolvedPartympl class and I decided to use it here because it is a simpler form of the for each loop and in this instance I don’t need to know what the index is of the element within the list as I am just invoking the isCurrent() method on the element to check to see if the involvedParty element is current, and then if yes, invoking the getPersonRef() method on the InvolvedParty element to determine if the personRef provided as a parameter to the method is equal to the personRef return and if so return true. This type of loop has been used a number of times during the project.

* + - 1. **Inheritance**

As you can see from the class diagram in Appendices C I have adopted the design pattern inheritance, this is to re-use code and make use of an important programming technique called polymorphism.

As outlined above in the descriptions of the classes I have created, I have created a super class called AgreementImpl, which hold generic fields and methods for an account, and I have then created TenancyImpl, LeaseImpl and ContractImpl which all extend AgreementImpl and therefore re-use the code of the AgreementImpl class.

Also by implementing inheritance here, it makes the system easier to evolve, for example if ‘MSc Properties’ was to expand and create a new Agreement, the system will be able to be amended to add an additional AgreementImpl subclass which can then make use of any methods that use polymorphism without any changes to the existing code, an example of a method which makes use of polymorphism us below.

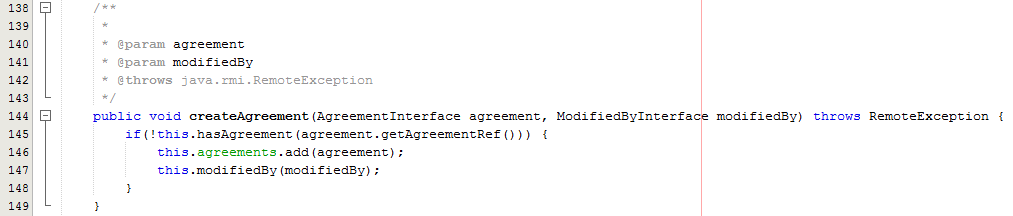


Fig x – Extract from Database, updateUserAgreements() method

As you can see from Fig x, the Office class has made use of polymorphism by having a List of agreements which can hold any type of agreement, whether it be a tenancy, lease or contract and within the createAgreement() method, you can see that the office class invokes Agreement.getAgreementRef() on any of the Agreement elements, again whether it be a tenancy, lease or contract.

I also used other design patterns to assist in developing this system for ‘MSc Properties’ however it is difficult for me to document all of these within the report.

* + 1. **Graphical User Interface**

Unfortunately, during the development process, I had some issues with building the graphical user interface (GUI), and although I could have decided to use a GUI builder that is provided by most integrated development environments (IDE) such as Netbeans, I wanted to build the GUI from scratch, writing all the code myself.



Fig x – Extract from LoginForm class, extract from layoutComponents()

As you can see from fig x, I was attempting to use the grid bag layout manager for the data entry forms (Person creation, Application creation, etc), this is because this layout manager provides flexibility to the way in which components such as buttons, labels etc, can be laid out on screen, as they are laid out in cells, with columns being any width, and rows being any height, providing greater flexibility for layout.

To implement grid bag layout I had to firstly set the layout of the panel in which my form is going to be within, by invoking setLayout() and passing as a parameter a new GridBagLayout object. I then had to declare and initialise a GridBagConstraints object, which is used to define the layout of any components added to the panel.

Once I have a GridBagConstrains object I then invoke methods from the GridBagConstraints class to set the constraints of any components added. The basic constraints are gridx() and gridy() which defines what position I am going to add a component on the screen, and gridwidth() and gridheight() determines what size the cell will take up on the screen.

Another concept I was adopting through the system, was the model-view-controller (MVC) architectural pattern, which I have explained was being used with the client package being the view, the Server class being the controller, and the rest of the server package classes being the model. However, I also adopt MVC within the GUI, with the ClientImpl being the model for the system, and a main frame GUI such as the HomeForm being the controller, and then panels of components, that have been added to the main frame which are the view such as a JTable or JTree.

The controller (main frame) would then deal with updating the view (panels of GUI components), as and when something happens in the model (ClientImpl class). The use of MVC within the GUI reduces the level of coupling between the model and the view, as the controller manages communication between the two.



Fig x – Extract from HomeForm class – declaration of ListPanel object

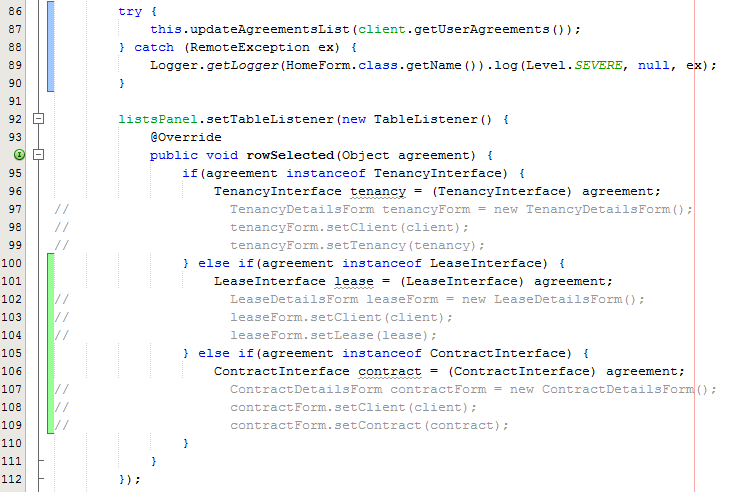


Fig x – Extract from HomeForm class – HomeForm constructor



Fig x – Extract from ListsPanel class, initialising TableListener field (action listener)



Fig x – Extract from ListsPanel class, setTableListener()

As you can see from fig x, fig x, fig x and fig x, I am ensuring the GUI makes use of MVC, by assigning the JPanels within any main frame with listeners, so if anything occurs within a panel (such as the listsPanel for the HomeForm shown above), instead of the listsPanel invoking a method from the HomeForm to notify the HomeForm of any change that has occurred within the panel, the panel is passed an action listener which listens to see if any action has been performed, and if so carries out a function within the main frame.

This ensures that the view (ListsPanel) does not know about the controller (HomeForm) and only interacts with the action listener that was passed to the ListsPanel by the HomeForm.

I have documented a large portion of the GUI I was attempting to develop for this project in Appendices G, along with annotations on what components and layout managers I was attempting to use to develop the GUI.

* 1. Iteration Cycle 2

For the second iteration of this project, I decided to implement the object relational mapping design pattern to store and manage the object-oriented objects in the database class of the system (‘MSc Properties’ business data), into the MySQL database I am going to create.

I need to make use of an object relational mapping design pattern because as you can see from the class diagram in Appendices D, the data stored in the Database class is non-scalar values, however as you can see from the enhanced entity-relationship diagram in Appendices E, the data stored in the MySQL database is scaler values and therefore the non-scaler values need a way to be broken down and stored in the MySQL database.

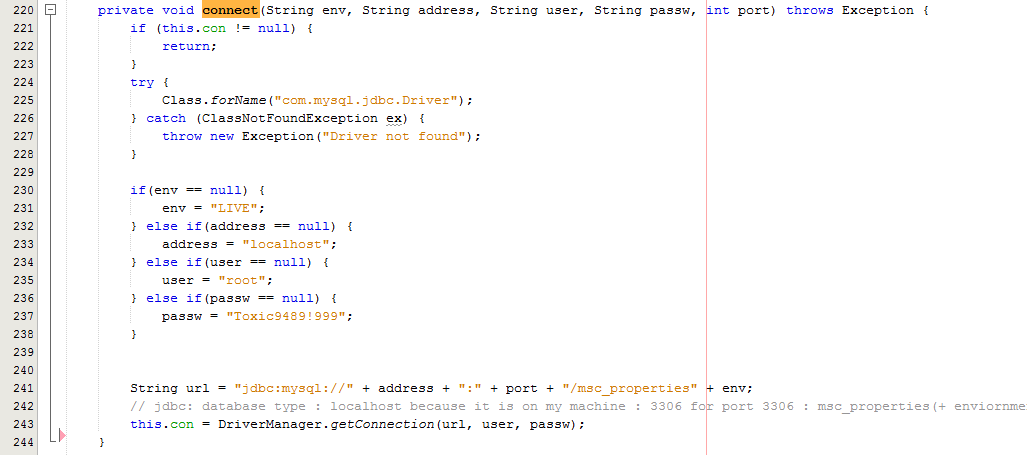
For the project, I had to create the database that was designed in the enhanced entity-relationship diagram using MySQL Workbench, and set up the settings for the database, such as primary keys, foreign keys, column data types and enforce referential integrity, which will ensure that any data stored in the database has to meet the rules set out by the relationships, meaning a foreign key value of a table, has to be present in the related table as a primary key.

Once the database was created I then had to use an application program interface (API) to allow the system to interact with the MySQL database. I need to use an API for the application and MySQL database because the two parts don’t know how to communicate with each other, and the API sets out how communication between these two take place. The API I will use for this project is Java Database Connectivity (JDBC) and is part of the Java Standard Edition platform. I am now going to take you through the steps taken to implement JDBC and set up communication between the system and the MySQL database.

* + 1. **Connecting to Database**

To implement a connection to the MySQL database from the Database class, I had to amend the Database constructor to pass MySQL database connection information (IP address, username, password, and port number), along with the environment the system is, i.e. LIVE, TRAIN or TEST.

Now the Database class has the connection information, the constructor invokes a local method called connect() and passes the information as parameters.



Firstly, I use the singleton pattern to ensure that the variable Connection con that is defined within the Database class is null, and if not then I don’t execute the main try statement, which creates the connection and assigns it to the con variable.

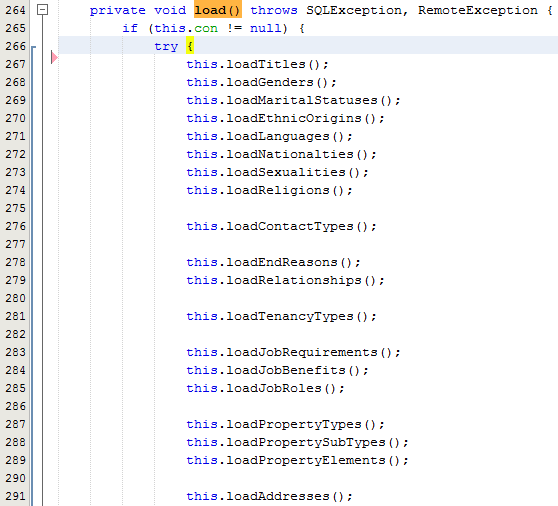
If there is no connection open already, we execute a try statement, which invokes the static method Class.forName(), providing the String value “com.mysql.jdbc.Driver” as a parameter. The Class.forName() method dynamically loads the JDBC driver which will enable the Database class to interact with the MySQL database. If the JDBC driver was successfully loaded without throwing any exceptions, then I check to ensure all of the information supplied as parameters are not null, if any value is null then I replace it with a value that suffices.

I then constructed a String value called url consisting of the API type, the database type, the IP address of the database, the port number and the database name, which for ‘MSc Properties’ consists of msc\_properties + the environment, for example msc\_propertiesLIVE. Once the string is constructed I then invoked the static method DriverManager.getConnection() and pass the String url, and the database username and password as parameters to the getConnection() method, the getConnection() method then returns a Connection object, which I assign to the con variable. I am then able to invoke methods on the con variable to interact with the MySQL database.

* + 1. **Loading System Data at Start-up**

To be able to load system data at system start up, was quite a tricky task as I had to ensure that no objects was loaded up prior to an object that the loaded object is dependent on was loaded, and also needed to ensure all system elements such as title codes, religion codes was loaded up first.

Once I had mapped out a flow in which I could load data from the MySQL database and create objects from that data without causing any issues with dependent data not being available, I was then able to create a load method which will create the system objects within the database class at system start up.



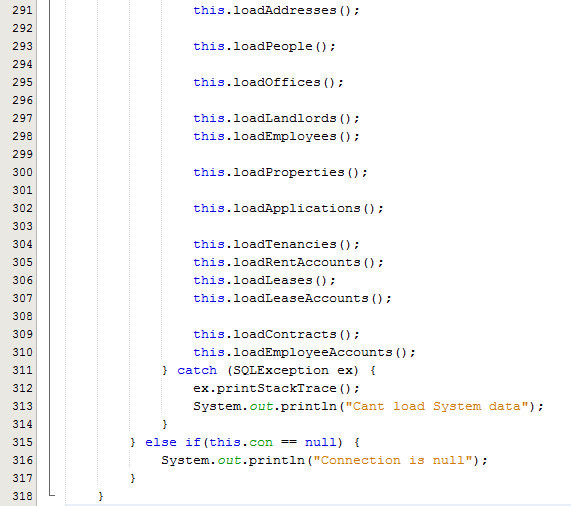


Fig x – Extract from Database class, load()

Once the above load method was created I then had to implement the individual load methods which will deal with loading sets of records from the MySQL database, create the objects and add them to the Lists within the Database class.



Fig x – Extract from Database class

The loadAddresses() method shows that firstly I had to create a String called sql, which will be the sql select statement I want to execute to retrieve the address records from the database. I then execute a try with resources statement, which declares one or more resources [], I declare a Statement variable called statement, and assign it the return value from invoking createStatement() on the con variable for the Database class within this try with resources statement.

Once I have the Statement variable initialized I am then able to invoke executeQuery() on the Statement variable and pass the String sql as a parameter, this will return a ResultSet object which contains the returned data from executing the select statement, the ResultSet object is then assigned to a ResultSet variable I declared called results.

Now I have the returned results I use a while loop with the condition being the return value of invoking next() on the ResultSet variable, which returns true if there is another record to return. So if there is another record in the ResultSet variable, I then invoke a get method to return a piece of data depending on the column name given as the methods parameter. There are a number of get methods such as getInt(), getDate, getString etc. to return all different data types.

Once all of the records information has been retrieved I then create the required object, in this example I had to create a Note object, which is a parameter for the creation of the Address object, once the objects have been created, I add them to the lists within the Database class. Once the method has finished I then need to close the statement by invoking close() on the Statement variable.

* + 1. **Creating Data, Updating and Deleting Data**

Although I am not going to talk about creating, updating and deleting records from the MySQL database in as much detail, as the tasks are similar I will show you a brief example of each.



Fig x – Extract from Database class, createAddress()

In Fig x, I am creating an insert statement, a PreparedStatement variable called insertStat, which means I can then add the information to the PreparedStatement through the use of set methods being invoked on the PreparedStatement and using the ? placeholder for the values I am going to supply. Once all of the information has been assigned using the set methods, the executeUpdate() method needs to be invoked and then the connection needs to be closed through close(). Once the connection is closed, I then add the objects to their respective lists within the database class.



Fig x – Extract from Database class, updateAddress()

In Fig x, I am creating an update statement, and as with the insert statement I use a PreparedStatement to supply the update values, use the set methods to set the values, executeUpdate and then close the connection.

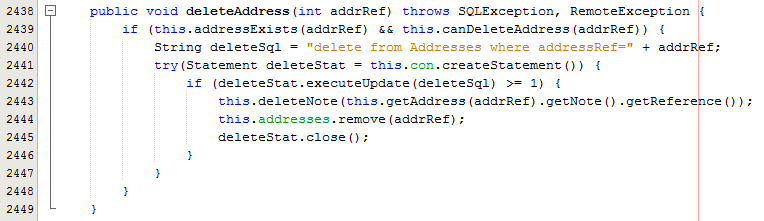


Fig x – Extract from Database class, deleteAddress()

In Fig x, I am creating an update statement, and as with the load methods I use a Statement, and just invoke the executeUpdate method on the statement and supply the deletSql String value as a parameter. I then remove the object from the List within the Database class and close the connection.

The importance of implementing the MySQL database is that if the system crashes or needs to be shut down (over the periods ‘MSc Properties’ is closed), there needs an external storage outside of the system to store the information of the system, and then when the system starts up, we are able to access this information as I have shown above to bring the state of the system back to what it was prior to shut down or system crash.

* 1. **Iteration Cycle 3**

For the third iteration of this project, I decided to implement the networking functionality, to enable a user of the system to have access to data and documents stored within the system, where the system is not necessarily local to them.

From the research undertaken, I decided to implement remote procedural call functionality, and as I decided to write the project code in Java, making use of the object-oriented concepts, the Java API, Remote Method Invocation (RMI) was the best decision for me as this enables the user in one Java Virtual Machine (JVM), which may or may not be remote to the server, to be able to invoke a method on a server object in another JVM.

For me to implement RMI there was a number of steps I needed to take, which are documented below.

* + 1. **Set up Server**

To implement RMI I needed to amend the ServerImpl class created in iteration 1, so that the ServerImpl class registers itself with the RMI registry so that clients are able to locate the server when they want to connect.

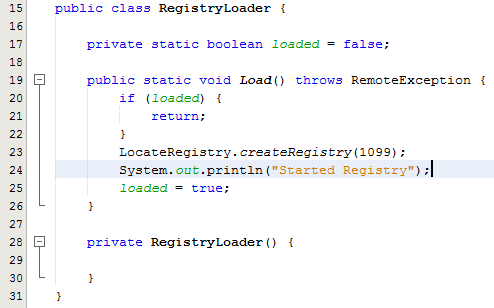


Fig x – RegistryLoader class

As you can see from Fig x, I decided to create a RegistryLoader class which deals with creating the registry, and setting it up on the local host to deal with requests on the specified port, to do this I invoked LocateRegistry.createRegistry() and supplied the port number that the server should communicate through. I used the singleton pattern to ensure that if the registry had already been created with the RMI registry then the server class is unable to try and register again.



I then amended the ServerImpl class to extend UnicastRemoteObject, which enables me to then export a Server instance of ServerImpl, which can then be registered with the RMI registry, and allow clients to retrieve the server stub to then invoke remote methods.

Due to design decisions which will be discussed later in this section, I also had to amend each class within the system model that was going to be made available to the client, to make them extend UnicastRemoteObject, however only the server stub needs to be registered with the RMI registry as explained earlier, the server will act as the controller, between the client package “View” and the rest of the server package “Model”.

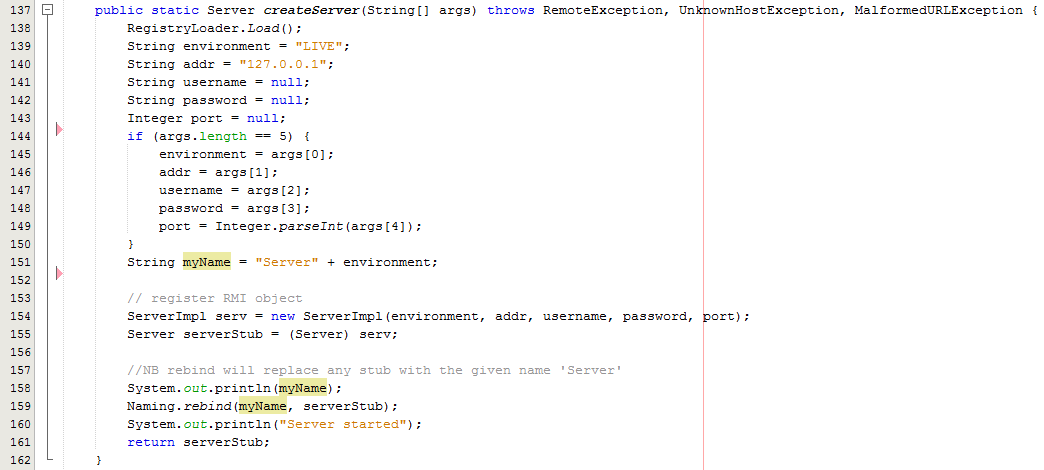


Fig x – Extract from ServerImpl – createServer()

I then amended the ServerImpl.createServer() method to invoke RegistryLoader.Load() method, which will invoke the RegistryLoader class, I then extract the information supplied from the client from the String array called args supplied as a parameter, if the String array has 5 elements then each element is supplied to the variables declared and some are then passed as parameters to invoke the ServerImpl constructor and create a new ServerImpl instance.

I then invoke the Naming.rebind method which deals with binding the specified name of my server to a new server stub, which is a Server (Remote) instance of the ServerImpl instance just created. The name of the server is Server + ‘environment’, (where environment is either, LIVE, TRAIN or TEST), which enables this server software to be run on 3 different hosts and act as a live, train or test environment for ‘MSc Properties’.



I also had to make any remote methods, or any methods dependent on a remote method for the server side classes, to throw a Remote Exception to the client invoking the method.

* + 1. **Set up Client**

Once the Server side coding was complete I then needed to create a client package, and as explained before, I already had a common package which consisted of any classes or interfaces which would be common between both the server side and client side packages, this meant that I firstly needed to add the common package into the newly created client package.

The first step I took was to create a ClientImpl class within the client package, and an interface for ClientImpl which I added to the common package, ClientImpl would then implement Client. I then needed to get the server stub that was registered with the RMI registry, which will be the object that the client invokes methods on to interact with the system.

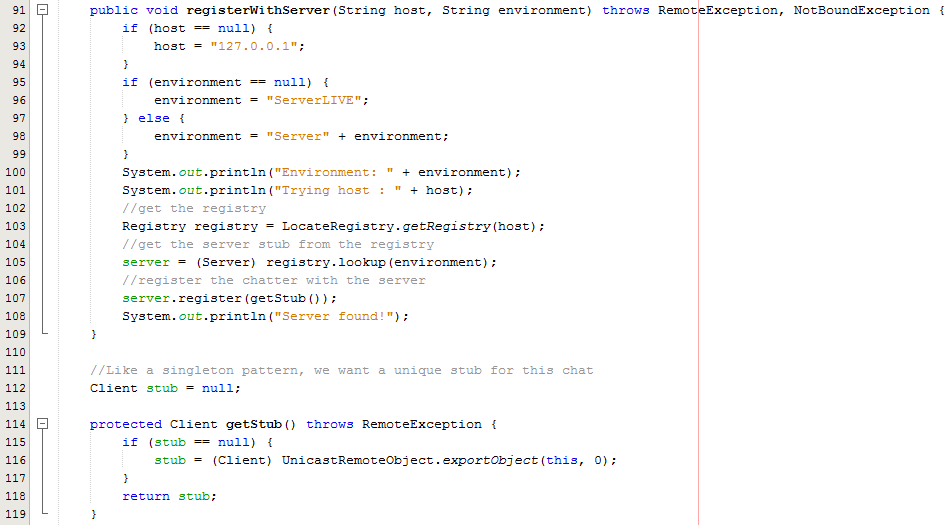


Fig x – Extract from ClientImpl – registerWithServer() and getStub()

To allow the Client to have a stub of the Server, I needed to get the Registry using the IP address of the Server by invoking the static method LocateRegistry.getRegistry(), and then invoke the lookup() method on the Registry object, passing the Server name as a parameter, which returns an instance of the Server, I then invoke register() on the returned server object, and pass a Client stub as a parameter, this Client stub will also be stored at the Server.

Once the Client has a stub of the server, the Client is then able to invoke any methods that is available through the Server interface.

* + 1. **Push vs Pull**

As outlined previously in the literature review carried out, there is two concepts for a distributed system, which outline the way a client and server interact with each other and initiate tasks, these are called the push model and the pull model.

For this project I decided to implement both forms of client server interaction, this is because when a client wants to perform an action, I decided to implement the pull model, this meant that clients of the MSc Properties system will request for the server to perform an action, so the client is pulling from the Server.

* 1. **Iteration Cycle 4**

To implement the document management functionality, I decided, instead of using a document management framework such as Apache JackRabbit, I would write my own document management framework that would deal with storing documents and versioning.

To do this I firstly created a Document class which stored a file and any previous versions of that file. I then had to implement methods both on the Client side and the Server side, that will convert a document into an array of bytes, this allows for the array of bytes to be passed between the client and server and then reconstructed at the other side, for the client or user to either view or save depending on whether a document is being uploaded to the server or downloaded by a client.

* + 1. **Uploading a document to the Server**

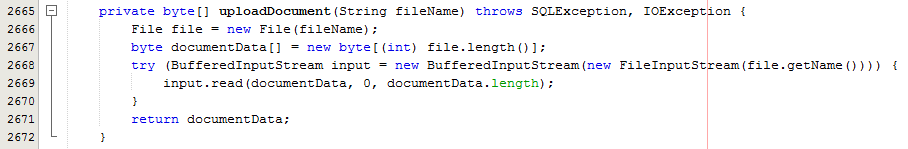


Fig x – Extract from ClientImpl – uploadDocument()

As you can see from Fig x I implemented a method that constructs an array of bytes from a given file. To do this I used a BufferedInputStream, and supply the constructor of the BufferedInputStream with a FileInputStream for the given file that is to be uploaded. I then invoke the read() method on the BufferedInputStream, supplying the empty array as the destination to read to, and zero and the document length as the boundaries for the read. I then return the array of bytes back to the invoking method.

The byte of arrays is then able to be passed to the client as a parameter along with the document name and some other information associated with the document, for the Server to then store the file.

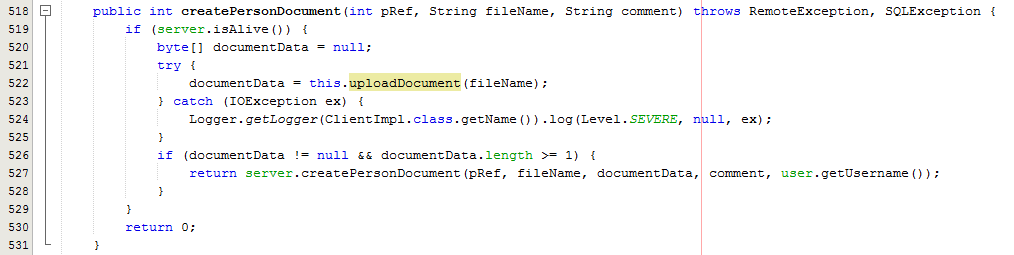


Fig x – Extract from ClientImpl class – createPersonDocument()

As you can see from Fig x, once the array of bytes has been created, I then check to see if the array of bytes (document data) has elements within the array, if so I then invoke a method on the Server which creates the file on the Server with the associated information.

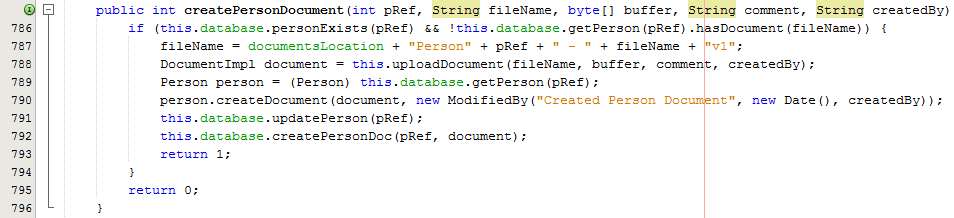


Fig x – Extract from ServerImpl class – createPersonDocument()

As you can see from Fig x, I firstly check to ensure the information supplied is valid, if so I then amend the filename to include the location the document will be stored at, along with some uniquely identifying information for the object the document belongs to, such as Person information, and then finally add the version number. I then invoke a uploadDocument() method which reconstructs the document as shown below and then return a Document object, which holds the filename and some other document information. I then update the database, and save the document file path to the database. The server now has a local copy of the file, which can then be accessed by any client of ‘MSc Properties’ system, by downloading a version of the document.

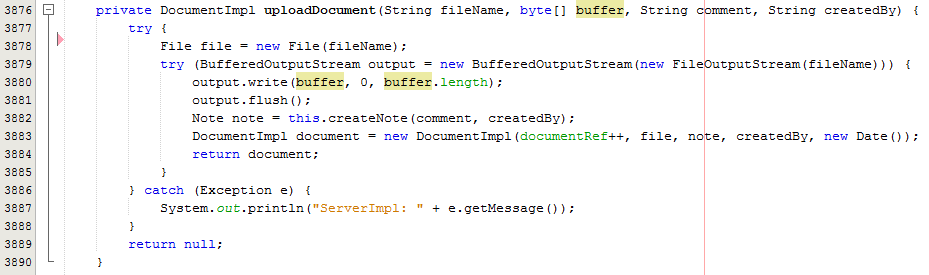


Fig x – Extract from ServerImpl class – uploadDocument()

As you can see from Fig x, I do the reverse to when I am converting the file into an array of bytes, and use a BufferedOutputStream, with a FileOutputStream for the file, as a parameter for the constructor of the BufferedOutputStream. I then write from the array of bytes called buffer, supplied as a parameter to the uploadDocument method, using the BufferedOutputStream.write method with the array of bytes as the source for the data, and 0 and the length of the array, as boundaries for the write method. This write method then writes the data to the file specified by the file name. I can then create a Document object, which will store the file name, along with the file path and some other bits of information associated with the document.

* + 1. **Downloading a document from the Server**

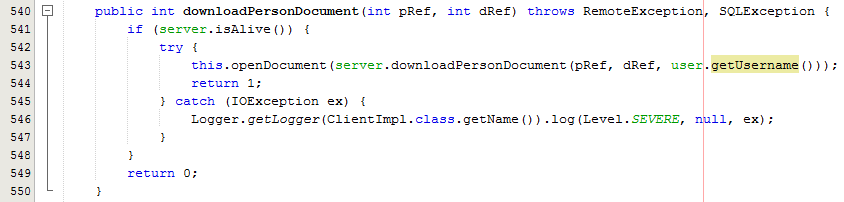


Fig x – Extract from ClientImpl class – downloadPersonDocument()

As you can see from Fig x, when a client invokes the downloadPersonDocument method it invokes the method openDocument() and supply it with the return value from the method Server.downloadPersonDocument() which would be the array of bytes for the document to be downloaded.

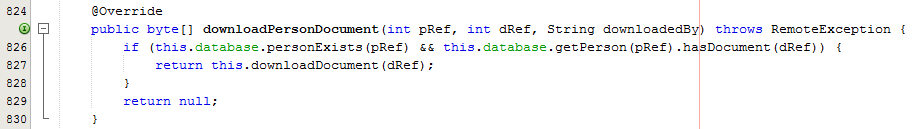


Fig x – Extract from ServerImpl class – downloadPersonDocument()

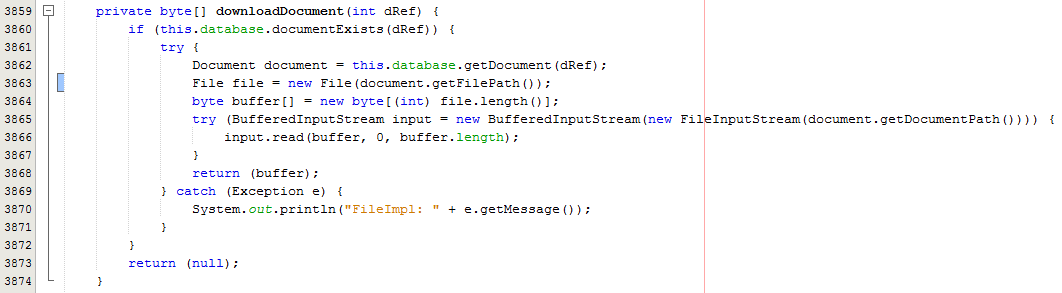


Fig x – Extract from ServerImpl class downloadDocument()

As you can see from Fig x and Fig x, The downloadPersonDocument() method just checks to see if the parameters supplied are valid, and then invokes a local method downloadDocument(), and the downloadDocument() method converts the file stored locally to the server, into an array of bytes in a similar fashion to how the Client does when the client is uploading a file to the server, using BufferedInputStream and FileInputStreams, and read the data from the file into the array of bytes. The array of bytes is then returned to the invoking method.

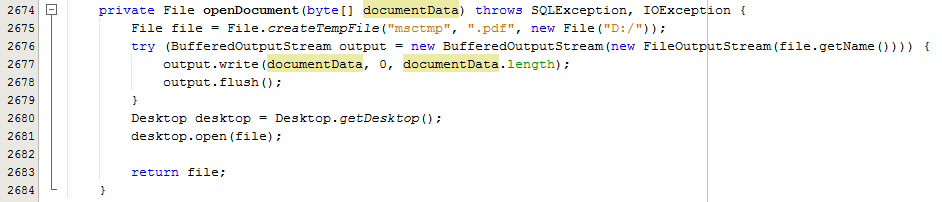


Fig x – Extract from ClientImpl class – openDocument()

Now the server has supplied the openDocument() method with the array of bytes for the file, as you can see from Fig x, I then have to reconstruct the file, in the same way the Server had to reconstruct the file when a client uploads a document to the file. To do this I again use a BufferedOutputStream, with a FileOutputStream, and then invoke the BufferedOutputStream.write method to write the data to a file.

As the client does not need to save a copy locally to them, I decided to use the File.createTempFile() method and supply it with a temporary name, this then allows for the file to be deleted once the client has finished with the file, however it also allows for the client to manually save a copy of the file if they do require.

Once the data from the array of bytes has been written to the temporary file specified by the file name, I then create a Desktop object, which is initialised with the Desktop.getDesktop() method, which returns the clients desktop. I then invoke the Desktop.open() method on the desktop object, and supply the temporary file I have just created. This then opens the temporary file, with the default application for the type of file that is being opened. This leaves the client actually viewing the file they selected to download.

* + 1. **Version Control**

To implement version control functionality, I decided to create a list of File objects within the document class, which enables me to add an updated file to the document object, and with each updated file that is added to the document object, the filename version increments by 1, to keep a track of which version the document is currently at.

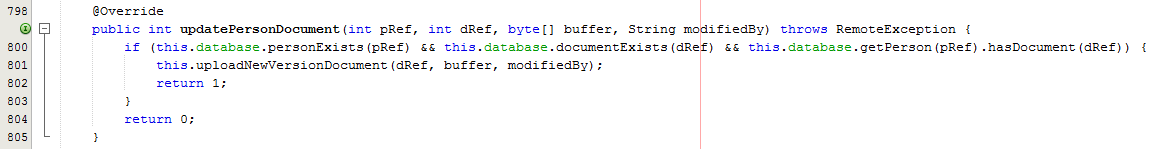


Fig x – Extract from ServerImpl class, updatePersonDocument()

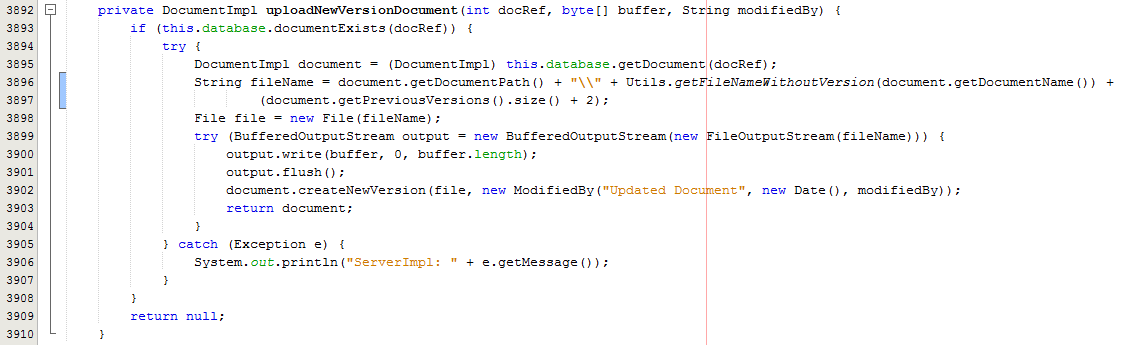


Fig x – Extract from ServerImpl

As you can see from Fig x and Fig x, when the client wants to update a document (create a new version of a file), the client invokes updateXDocument(), with X being a specific object such as Person, and again the client just supplies an array of bytes for the new version of the file, this is then constructed back into the file in the same way as previously shown, but instead of using the same file name as was previously used when the document was first created I increment the version number by 1.

This is done by invoking a static method from my own Utils class which extracts the filename without the version number or the extension and uses the list of files size from the document object to get the new version number and then reconstruct the filename with the new version number.

A client of ‘MSc Properties’ is then able to view the most current version of the document, but can also see a list of previous version by invoking the Document.getPreviousVerions() method.

* 1. **Iteration Cycle 5**

For the next iteration I decided to implement the observable pattern, which will be used to update a client of ‘MSc Properties’ graphical user interface (GUI) when an agreement or rent account is updated, without the ClientImpl class from the Client package actually knowing about the GUI.

I decided to implement the push model of the observable pattern, instead of the pull model that the client server data exchange is predominantly based on (as explained in iteration cycle 3 – push vs pull), this is because the pull variant can be quite costly, because each observer will invoke the method to pull through the current state even if there have not been any changes, whereas the push variant only updates observers when an update has occurred. This selection was best for this type of implementation because there will not be a frequent number of updates to ‘MSc Properties’ agreements or rent accounts so it is unlikely the observers will need to be updated all the time.



Fig x – Extract from ClientImpl, class header

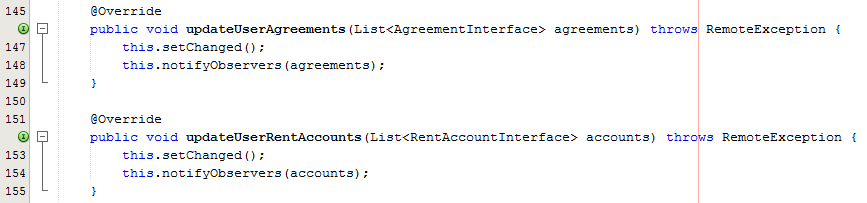


Fig x – Extract from ClientImpl class, updateUserAgreements() - updateUserRentAccounts()

As you can see from fig x and fig x, to do this I first had to make the ClientImpl class extend Observable, which meant that I had to create a method which notifies any observers when there is a change.

The methods that notify the Observers (GUI) when any changes to either agreements or rent accounts are updateUserAgreements() and updateUserRentAccounts() and both of these methods invoke two methods inherited from the Observable class, setChanged() and notifyObservers() which will notify the list of observers, that there has been a change to the object being observed (ClientImpl), and then passes the updated object as a parameter of the notifyObservers() method to the observers (GUI).

Now I have to amend the home screen, which is the GUI that will be the observer, and need to update whenever the observable notifies of any state change.



Fig x – Extract from HomeForm class (GUI), class header

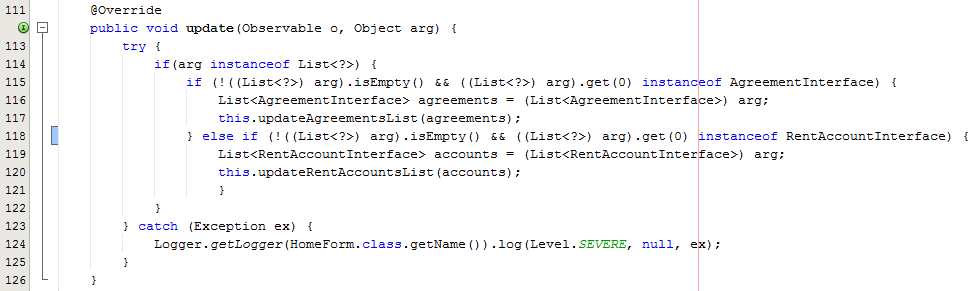


Fig x – Extract from HomeForm class (GUI), update()

As you can see from fig x and fig x, the HomeForm class implements Observer, which also means as it is a GUI, it is able to still extend JFrame, but also means that it then needs to provide an implementation for the update method. Which as explained before, will be invoked by the Observable class, when the Observer invokes setChanged() and then notifyObservers(). As you can see from fig x, the updated object is passed as a parameter. However, because it is passed as an Object I need to check if the object is an instance of the required object, and because the object passed should be a list, I first need to check to see if the Object is instance of List. But because of Type Erasure the compiler at run time does not know the type of object within a list, so I am unable to test if the list has the correct type of elements without actually obtaining an element from the list and checking the elements type.

So if the object is of type List, I then need to check if the list is empty and if not then actually get an element out of the list and test the type of the object is either instance of Agreement or instance of Rent Account.

I then invoke an update method which will amend the GUI display to reflect the change that has occurred to the Observable object.

The last part of the implementation of the Observable pattern (although it is actually now fully implemented), is that the controller must invoke ClientImpl.updateAgreements() or ClientImpl.updateRentAccounts() and again this is one of the only implementations of the push data exchange model, because the server is actually pushing a change to the client, instead of the client pulling “requesting” data from the server.



Fig x – Extract from ServerImpl class updateUserAgreements()



Fig x – Extract from ServerImpl class updateUserRentAccounts()

As you can see from fig x and fig x, each of these methods prepare the wist which will be the updated list (object to be passed via the notifyObservers method from ClientImpl class) after any changes have occurred. Once the list is created and all required elements have been added to the list, I then go through a list of clients checking to see if client is still alive, and if so if the client needs to receive the update (only send out update to clients of that office), and if the client is then I invoke Client.updateUserRentAccounts() or Client.updateUserAgreements(), and pass the updated list as a parameter to the method.

This then makes the Observable object the ClientImpl invoke the setChanged() and notifyObservers methods discussed earlier, which invokes the Observer objects update method. The observer pattern then allows the system for ‘MSc Properties’ to ensure that the clients home form is always updated, but also does not send unnecessary updates to all clients that don’t need the update.

* 1. **Iteration Cycle 6**

My final iteration cycle was to automate a number of tasks that have to be carried out by ‘MSc Properties’ officers. To do this I decided to use the Java class TimerTask, and created a class called TaskGenerator, which extended TimerTask class from the Java util package.

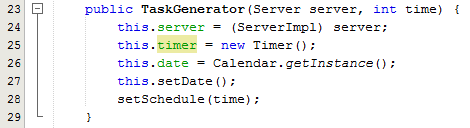


Fig x – Extract from TaskGenerator, TaskGenerator constructor

As you can see from fig x, I pass a server object and an integer time value, which is the time in milliseconds between each task that is to be generated.



Fig x - Extract from ServerImpl class, create a TaskGenerator variable



Fig x – Extract from ServerImpl constrctor, initialize TaskGenerator variable

As you can see from fig x and fig x, I create a TaskGenerator variable pass it this server instance and 86,400,000 (the number of milliseconds within a day), which then creates a TaskGenerator object that will run every 24 hours.

As you can see from fig x, I then initialise the timer variable with a new Timer instance, I create a new Calendar instance with Calender.getInstance() and then invoke local methods setDate() and setSchedule() where I pass the time in milliseconds to the setSchedule method as a parameter.

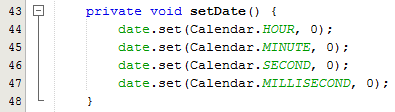


Fig x – Extract from TaskGenerator class, setDate()

As you can see from fig x, the setDate() method invoked within the TaskGenerator constructor just sets the time part of the Calendar variable called date to midnight, this will be used as the benchmark for when any tasks should be run, as it would be best to run any jobs at midnight as this is outside ‘MSc Properties’ business hours.



Fig x – Extract from TaskGenerator class, setSchedule()

As you can see from fig x, the setSchedule() method invoked within the TaskGenerator constructor invokes the Timer.schedule() method, and I supply this instance of TaskGenerator (which will be the object that will then do something when the timer has reached midnight), and the Date object returned from invoking Calendar.getTime() on Calendar variable date, and lastly the time in milliseconds between each task to be generated. Once the timer reaches midnight it will then invoke the method run() which I have had to override within the TaskGenerator class.

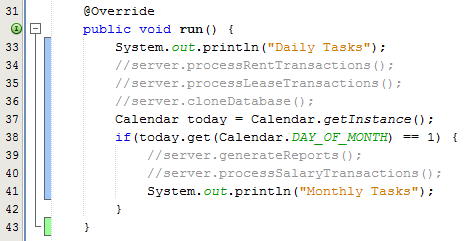


Fig x – Extract from TaskGenerator class, run()

As previously explained I had to override the method run() from TimerTask class, this is because I have extended the TimerTask class.

As you can see from fig x, within the run method I am carrying out a number of daily transactions such as Server.processTransactions() and also some monthly transactions such as Server.generateReports() which will automate tasks like creating rent transactions or generating monthly reports.

This will mean that an ‘MSc Properties’ employee will not have to manually create rent transactions for each tenancy every month (the start date of the tenancy determines when the rent is charged to the rent account), or the manager will not have to generate monthly reports each month. The TaskGenerator class will also be used to clone the LIVE database to the CLONE database, as a back-up for the LIVE data within the MySQL database.

* 1. Testing

As explained previously in iteration cycle 1, I carried out unit testing and system testing during each iteration cycle, but I have decided to document it all within this Testing section of the report.

1. Conclusion

I believe that overall this project was a success, this is because although I did not achieve all of the advanced objectives I set out for the project, and although I believe with further resources I may have been able to achieve the project objectives with higher quality, I did successively achieve a large portion of both the core and the advanced project objectives.

Of the core objectives, I was unable to fully achieve the following, ensuring **all** resources were available (not being able to install Apache to a Windows machine, meaning I was not able to create a web server, nor was I able to install Bugzilla, meaning I had to use Excel to record any bugs within a Bugs List), writing **all** test scripts, carry out **all** unit and system tests, and finally develop a user manual.

Of the advanced objectives, I was unable to fully achieve the following, develop a website, and develop a password reset facility.

Also although I believe the project was a success, the project plan I defined earlier in the project was not very realistic with regards to time allocation for tasks, and although I believe this was because of the lack of experience I have with developing systems of this size, it resulted in me allocating not enough time for tasks, which then had a knock on effect on other tasks when tasks overran.

However, although my project plan was not realistic I believe I managed the project well through frequent highlight reports (which highlights progress, risks etc.) at Appendices F, and was important in me realising early in the development that there was a major risk of not achieving all goals due to a lack of time, and making decisions with regards to design documents produced and functionality I was going to implement to allow me to achieve a large portion of the project aim and objectives.

The final area which I believe I did not do so well in is the development of the graphical user interface, as although I had designed a number of screens, and understood the theory to a certain extent on what was trying to achieve, when I was actually attempting to develop the GUI, the alignment of components on the screen was not correct, and because this is a project that will need to display quite a large amount information to the client, I needed to insure that the components were laid out correctly, to allow for the Client to understand the information, for example labels for field entry boxes, would need to be aligned correctly for an ‘MSc Properties’ employee to be able to use a creation form correctly.

Another area I believe was a success was the literature review, as this enabled me to get different views of different experts within Computer Science, and which assisted me with design and implementation choices made during the project.

Lastly my choices of methodology overall were good, and I believe may have only been bettered in certain areas. This is because the choice of language I decided to code my system in, Java, was best this is because it allows compiled code to run on all platforms, meaning that the code is usable across a wide range of devices, and as ‘MSc Properties’ may have different computer set ups at different sites, it allows for the system to still work on the different set ups.

The choice of Remote Method Invocation (RMI) as networking functionality, was the best because I was coding in Java and as explained before, RMI allows for objects to be passed as values, and therefore behaviour of an object can be passed across a network from the server to the client, instead of just primitive values such as integers. Also RMI makes it easier for the programmer as the programmer does not need to manage the sockets, threads and serialization of objects, making the development of the system easier.

1. **Evaluation**

As I outlined in the conclusion, I believe that overall the project was a success, however there is a few decisions I would have made differently if I was to complete the project again.

The first change to the project would be to set out a realistic project plan from the offset, which would result in a more manageable project, as I would hopefully not run over allocated time for tasks, meaning an amendment to documents I wanted to produce and functionality implemented.

The second change I would make to the project is to ensure that I had a Linux operating system available to me, because from research, I have found that this is the best operating system to manage a web server with, and also it would have allowed me to install Bugzilla a lot easier which in turn would have meant that I had a better bug tracking tool than what I decided to use due to not being able to install Bugzilla on.

Bibliography

References

**Further Reading**