Analysis of deployment techniques for web-based applications in SMEs

NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

Dissertation submitted to the Department of Computer Science, National University of Ireland, Maynooth in fulfilment of the requirements for the

M.Sc. in Computer Science (Software Engineering)

Cathal Browne, B.Sc.

January 2011

Work Placement Thesis

Supervisors: Patrick Marshall, Dr. Susan Bergin

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Master of Science in Software Engineering, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: ______________________________

Cathal Browne

Date: ______________________________

Acknowledgements

I would like to firstly thank my supervisors Susan and Patrick for all their time and for sharing their experience and knowledge. Without them I would still be stuck on writing the title.

I would like to express my gratitude to George and Niall in Topfloor. Firstly thank you for giving me the opportunity to work in a company that has given me a huge learning experience and many fun times. Secondly thank you for the support and encouragement during this project and during the writing of this dissertation.

I would like to thank Hana for her help with graphics and illustration software which produced the images and graphs for this dissertation.

Lastly I would like to thank my family for listening to my complaints when I got stuck during the research and writing and also for being supportive and understanding for all the missed visits since I started the masters course.

Abstract

The Internet is no longer just a source for accessing information; it has become a valuable medium for social networking and software services. Web-browsers can now access entire software systems available online to provide the user with a range of services. The concept of software as a service(SAAS) was born out of this. The number of development techniques and frameworks for such web-applications has grown rapidly and much research and development has been carried out on advancing the capability of web scripting languages and web browsers. However a key part of the life-cycle of web-applications that has not received adequate attention is deployment.

The deployment techniques chosen to deploy a web application can have a serious affect on the cost of maintenance and the quality of service for the end user. A SAAS modelled web application attempts to emulate a desktop software package experience. If a deployment process affects the availability and quality of service of a web-application then the core concept of this model is broken.

This dissertation identifies approaches to designing a deployment process and the aspects that influence the quality of a deployment technique. A survey was circulated to a number of Irish small to medium sized enterprises (SME) that develop web-based software. The survey shows an overview of multiple deployment processes used by these SMEs. Using this information, along with a review of the available literature and a detailed case study of a typical SME deploying SAAS based products, the dissertation provides a critical analysis and evaluation of the current deployment techniques being used.

Table of Contents

Chapter 1: Introduction

1.1 Background

As part of an M.Sc. in Software Engineering, a six month work placement had to be completed. In conjunction with the work placement, a software engineering project was undertaken and a dissertation written. There were two options for projects; a research project and a work placement related project. This dissertation is based on gained experiences and encountered problems in the work placement.

1.2 Work Placement

The six month work placement was completed in a small software company called Topfloor[1]. Topfloor is a small/medium enterprise(SME), with 3 employees and 2 software packages. The company specialises in web-based software for property management. For the last two years running, 2009 and 2010, Topfloor has been listed in the Deloitte Technology Fast 50 awards[2][3]. Topfloor produces two software packages, Letman and Blockman.

Letman allows agents to manage the renting of properties. Letman's basic features include, collecting rents and keeping record of landlords, tenants and leases.

Blockman allows agents to manage the running of an apartment block. Blockman's basic features include, issuing and collecting management fees and keeping record of all units in an apartment block and the unit owners. Blockman also manages events, for example, AGMs.

The advantage of working in a small company is that one gets to be involved in all areas of software development. The work placement incorporated the roles of user-interface designer, developer and system administrator. The responsibilities included Java servlet development, user interface development(HTML, CSS and Javascript) and server side scripting for taking database backups and ensuring/monitoring server uptime. One of the main achievements in the company during the six month work placement was the improvements made to the deployment of the software.

When the placement started, there were still customers with small servers in their offices running the software. This made it very difficult for Topfloor to deploy updates and in some cases no updates were made to deployed software. Over the duration of the work placement, these instances of the software were moved to a central server hosted by Topfloor. The Topfloor servers were upgraded to support this extra load. The upgrades included virtualisation, server load balancing, and caching.

1.3 Importance of high quality deployment

"Simply put, a system's deployment architecture is the allocation of the system's software components (and connectors) to its hardware hosts" [4].

Software deployment can be complex for small to large scale software systems. Prior to software release, developers need to consider two important factors, quality of service (QoS) and quality of manageability (QoM). QoS affects the users ability to use the software as intended. QoM allows a system administrator to efficiently install and maintain the software.

QoS is an important factor in the success of a software product. The different dimensions of QoS include latency, availability, durability, reliability, security, fault-tolerance, survivability, dependability, scalability and heterogeneity [4]. The level of QoS is closely dependant on the deployment of the software.

Managing a software deployment dominates system administration costs and configuring a software deployment is a major source of errors in system deployment[5]. A low QoM increases the costs of maintaining software and increases the chance of errors in the system. The introduction of errors and costly administration of software can cause disruption to a service and therefore reduce the QoS for a user.

1.4 Evolution of web-based software

When the Internet became available to the public in 1994/1995 it was just a collection of static pages. Hypertext was created to allow these static documents to link each other, rather than having to memorise the address of each page. It allowed the widespread creation of the connected web, as it is now known, the World Wide Web (WWW). In the space of a decade the WWW has evolved from these static pages to a powerful platform for application development and deployment[7].

Even with this evolution of the WWW, the fundamental building blocks have remained similar. These building blocks have been extended to facilitate the modern requirements of the WWW. The WWW is built on a simple request/reply, client/server model. The protocol in which the client and server communicate is the HyperText Transfer Protocol (HTTP) [8]. The client identifies a document on a server using a naming system called Uniform Resource Locator (URL)[9]. The server returns the information to the web browser with formatting instructions. The markup language HyperText Markup Language (HTML) instructs the web browser how to display the information. The HTTP protocol has been enhanced over the years to facilitate more efficient practices and encryption, using the Hypertext Transfer Protocol Secure (HTTPS[10]). A URL identifies the server by IP address and then navigates to the location of the document on the server. Domain Name System (DNS) is used to replace the IP address with a more convenient text based name in URLs. HTML has been extended numerous times over the years. Extensible HyperText Markup Language (XHTML) is an extensible markup language (XML) compliant HTML. It was created to make a standardised and dominant format of HTML. Recent additions to HTML are seen in HTML5. HTML5 contains extensions for a more multimedia and socially based WWW[11]. For example it allows the embedding of video media without the use of third party plugins. HTML was further enhanced with the use of Cascading Style Sheets (CSS) styling sheets. It allows the creation of more appealing designs for user interfaces.

"A first key observation was that the address that was considered to be a page of data on the server could in fact refer to a program that could be executed on the server and its results returned to the client"[7]

This quote describes the point at which the WWW changed from being a repository of static pages to hosting dynamic applications. Another realisation that aided this was that the web browser could be a universal user interface, which would dramatically save on development costs. One of the modern names for these dynamic web applications is the Web2.0 era. Web2.0 applications invite user participation for example social networks. They also try to emulate the user experience and responsiveness of a desktop application. This has become possible with the combination of client side scripting with JavaScript and a server side scripting such as PHP or Ruby in a standard now called Asynchronous JavaScript and XML (AJAX). AJAX allows communication between the web browser and server without invocation from the user (i.e. the user does not need to click a link or a button). This allows the web application to anticipate what information the user requires and to download it asynchronously. This gives the appearance of a very low-latency application.

Another factor contributing to the popularity of web-based software is the cost savings in deployment. Developers have the ability to release a new version of the software onto one location only. Instantly every user has access to the newest version of the software. The laborious and expensive task of traditional desktop-based deployment is now avoided. The developer is no longer concerned with notifying users of an update and dealing with the problems of multiple versions of the software in distribution. The quality of service (QoS) for the user is improved. Updates to features and software system patches can be deployed to a running web application as soon as they are ready for release. This allows developers to concentrate more on the development of the software rather than the deployment techniques used. A software company could also see the use of a web browser as a way to reduce training costs[12]. The web has become so common in modern life, that many of the potential users have already learned the skills necessary to use a web application.

1.5 Software As A Service

In the software as a service(SAAS) model, the software application or service is deployed in a central location. Users access the application over a network such as the Internet or a company Intranet. Users of the software “rent” or “subscribe” to gain access to the software rather than the traditional “purchasing” of the software and installing the system. Other names commonly used for this model are[13]:

- ASP – Application service provider.
- AIP – Application infrastructure providers.
- IBS – Internet business service.
- BSP – Business service provider.
- SSP – Solutions service provider.

This dissertation will concentrate mostly on the deployment process involved for software developed using the SAAS model. This model provides the potential for creating full software suites for distribution on the Internet. Topfloor has embraced this model as the foundation of their entire business plan.

1.6 Web-based software deployment in small companies

In the work placement company, Topfloor, it was difficult to determine if the best deployment techniques were in use. There was a lack of suitable examples for comparison or documentation for evaluation. Many problems and challenges were encountered. When planning the deployment process, considerations had to be made for taking backups of the customer data. While web based software simplifies the task of issuing updates compared to traditional desktop software, there are still many questions that need to be addressed when issuing updates:

- Is there forced downtime to issue an update?
- Should each installation be updated one by one or can there be a shared resource common to all installations?
- What time of the day should the updates occur to allow for the least amount of disruption to users?

- How can the deployment process be planned so that updates can be easily automated?
- If there is a bug in a new update, how can the deployment be rolled back quickly?

Without any methods for comparison and evaluation, is not possible to verify if these issues were addressed adequately. Other challenges that were faced in deployment are the installation and deinstallation of the software. Deinstallation can also be referred to as the removal of an instance of the software or an installation. Each customer has their own installation, so the deployment process needed to facilitate installation with minimal configuration. The chances of introducing errors increases with the amount of configuration needed for an installation.

In an SME like Topfloor, deployment is never an individual's primary responsibility. The individual issuing the deployment process may also be concerned with a development or support issue, so there is still need for more automation in the deployment. Manual deployment allows for and contributes to the introduction of errors.

1.7 Dissertation Goals

The goals of this dissertation are:

- Determine the factors that influence the quality of a deployment process for web-based software.
- Gather information on current deployment practices in use by Irish SMEs.
- Analyse and critique current deployment practices in use by Irish SMEs.
- Analyse Topfloor's deployment process and make recommendations.
- Identify areas for further research into web-based deployment.

Section 1.8 describes how these goals are addressed throughout the chapters in this dissertation.

1.8 Approach to dissertation

As part of this dissertation a web-based survey was prepared for small to medium sized enterprises(SME) to complete. The survey collected data to provide an overview of an SME's deployment setup, their hosting infrastructure and specific details of development factors that significantly affect the deployment process. With this data, a critical analysis can be made of factors that lead to the design of a deployment process. Additionally comparisons and evaluations can be made with the deployment process used in Topfloor.

The remainder of this dissertation is organised as follows:

Chapter 2 describes the background to web-based software and the software development life cycle, with specific reference to deployment. It builds on the information already provided in this introductory chapter. It also highlights the challenges that need to be addressed during the deployment stage of the software life cycle. It highlights a research gap and a need for further research in web-based software deployment.

Chapter 3 discusses and documents the software products produced by the work placement company, Topfloor. It briefly describes the software design and the customer background. It details in depth the deployment processes used in Topfloor for the installation, update and deinstallation of Topfloor's software packages.

Chapter 4 describes the design process used to create the survey. It specifies the target data that is hoped to be gained from each of the questions. It contains predictions and assumptions that were made while the survey was being created. It also provides detailed graphs and explanations of the aggregated results of the survey.

Chapter 5 provides a critical review of the data obtained from the survey. A critical analysis is made based on the background obtained from the research in Chapter 2 and the existing deployment processes in Topfloor detailed in Chapter 3.

Chapter 6 validates the relevance of the research and that the initial research goals of this dissertation were met. It also highlights the directions for further research and justifies the need for further research in the area of web-based software deployment.

Chapter 2: Background

2.1 Software engineering

Software engineering can be dated back to the 1960s and came about to solve the problem of the high rate of failure with software projects[14]. Software engineering grew as a response to this "software crisis"[15]. The consequences of the software crisis led to problems such as:

- Software projects running over-time and over-budget.
- Software projects did not meet their priority requirements.
- Software was of low quality and inefficient.
- Software projects ended up in failure and never being deployed.

While tools and methodologies were created and documented, there was no single software engineering solution that could fit all scenarios. There were debates between research groups over claims of silver bullet or "one methodology solves all" solutions. However, even today, there is no universally accepted software engineering methodology to solve all scenarios[16].

2.2 Development architectures and methodologies

Three important areas to improve the quality of software and the development process were software engineering methodologies, software design patterns and software architectures. A software engineering methodology refers to the framework that is used to structure, plan, and control the process of developing an information system[17]. It is also referred to as a process of developing a software system. Methodologies included:

- Waterfall model.
- Spiral model.
- Agile.
- Extreme.

A software design pattern is a template of a reusable solution that models a commonly occurring problem in software design. They are usually specified abstractly so they can be implemented in any programming language. Software design patterns are usually object oriented based. Figure 2.1 illustrates a number of design patterns and their relationships.

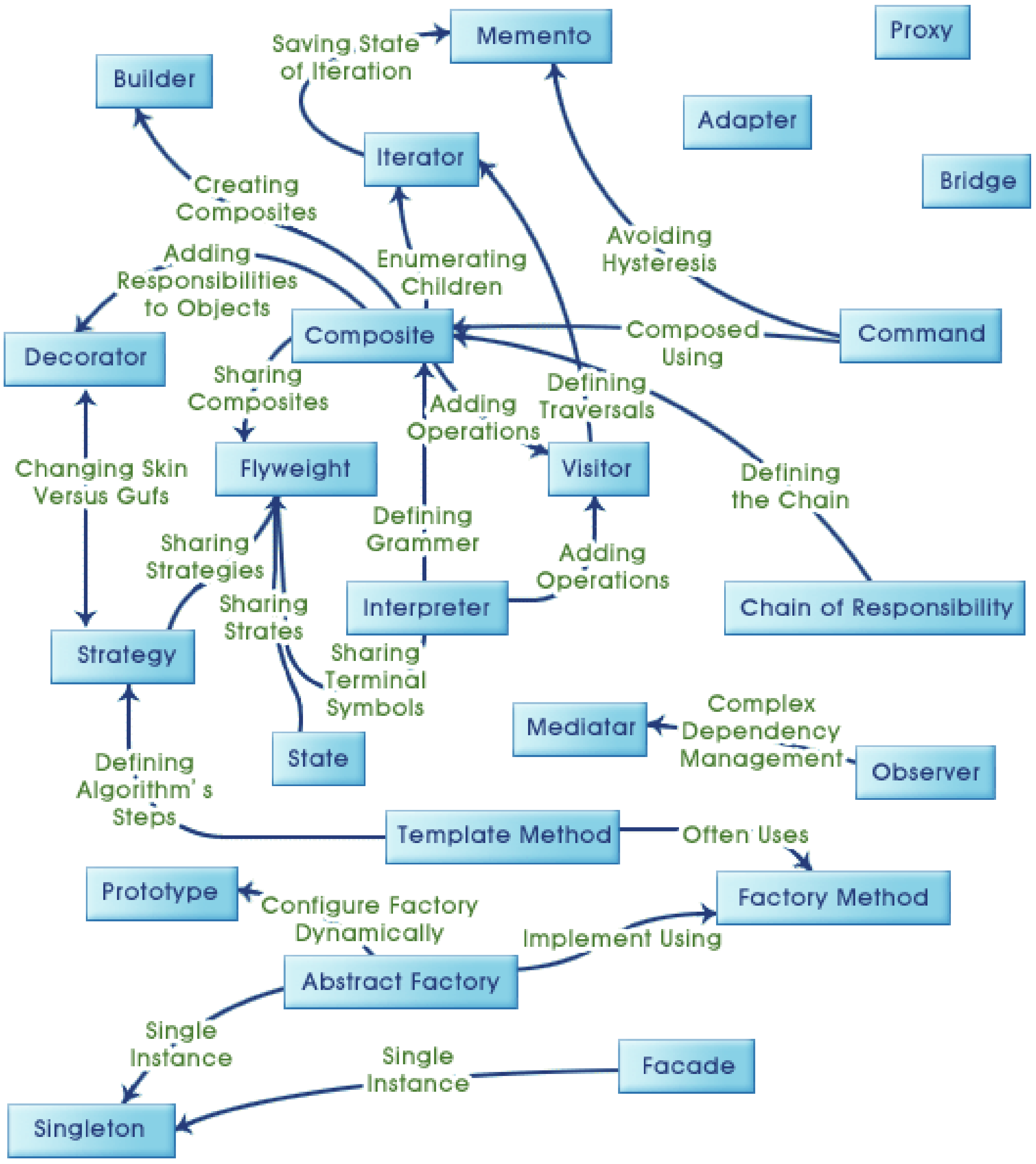

Figure 2.1 – *Design patterns and their relationships*

Software architectures are at a higher level of abstraction than design patterns. Design patterns provide a template for a problem within a software system. Architectures provide a template for the entire software system. Common architectures used today are:

- Client - Server Model (also known as 2-tier, n-tier, peer-to-peer).
- Model,View, Controller (MVC).

2.3 Client – server model

The server component provides a service to one or more clients. The client initiates the service by sending a request to the server. The server responds with the result of the request. The behaviour of client-server architecture can be described in a sequence diagram from the unified modelling language(UML). Figure 2.2 illustrates some basic server-client traffic.

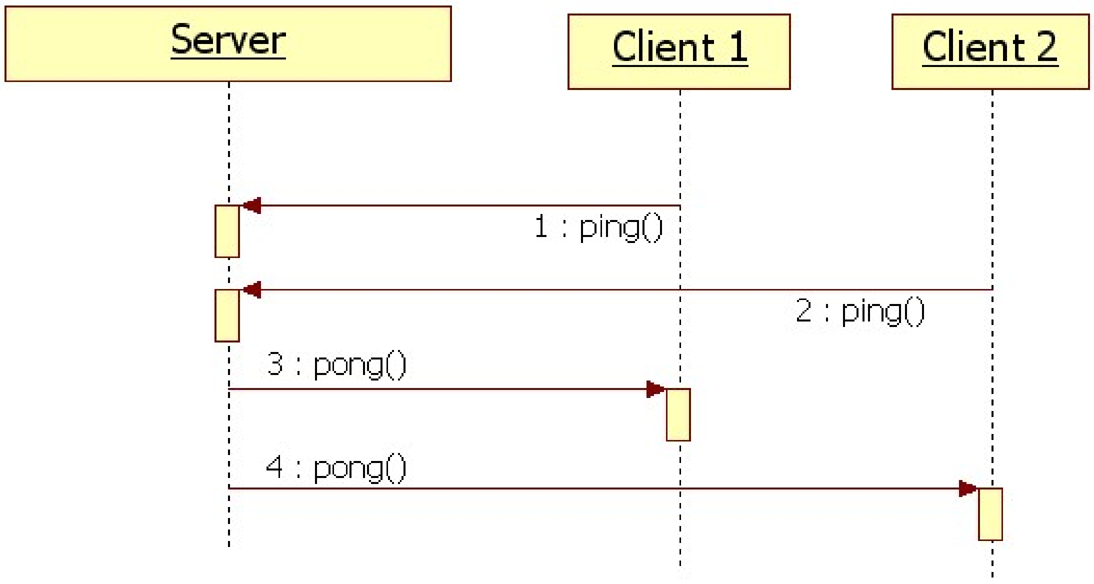

Figure 2.2 – *server-client sequence diagram modelling requests*

In web-based software HTTP is used as the communication protocol in the client-server architecture. HTTP specifies nine request methods [8], but only the two commonly used methods; GET and POST are discussed here. A GET request is recommended as the method for passing parameters when requesting data. A POST request is recommended when data is to be submitted for processing. The server uses status codes in the first line of the response to inform the client how to act. Below is a list of the most commonly seen status codes:

- **200** – OK, standard response for a successful HTTP request.
- **301** – Moved permanently, this informs the browser to redirect to a new location.
- **403** – Forbidden, the request was valid but the server is refusing to respond due to insufficient permissions held by the client.
- **404** – Not Found, the requested resource does not exist.

The status code is then followed by further headers that describe the response content. In web applications the response is usually a HTML document. The web-server can serve any kind of file or content in the response and the HTTP headers inform the client of what the content is. This is used to embed images and media in websites. It is also used to transmit data in other formats such as XML. Figure 2.3 illustrates communication between a web-server and a client.

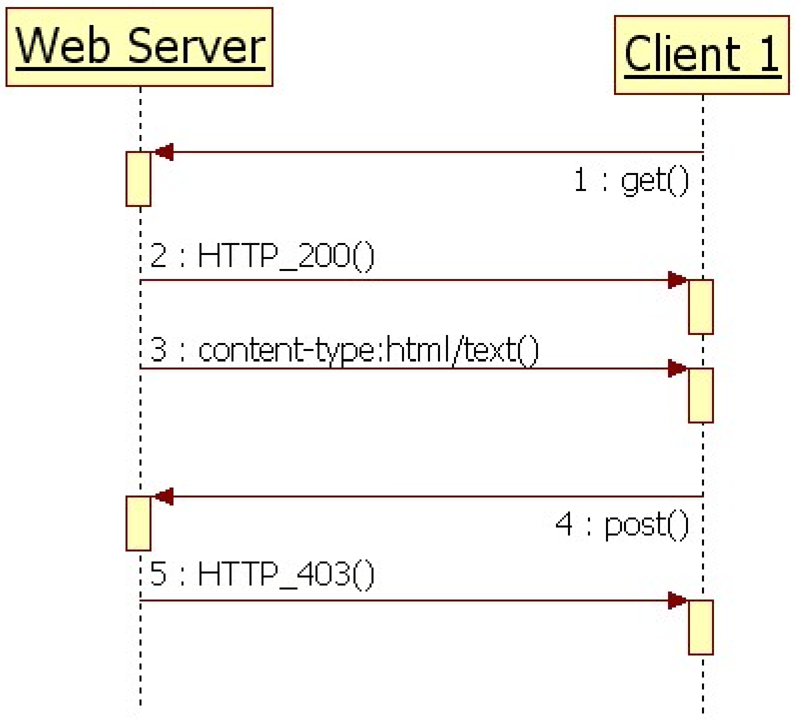

Figure 2.3 – *sequence diagram modelling HTTP requests between web-server and client*

2.4 Evolution of the Internet

"the Internet has become a platform of choice for a large number of ever-more sophisticated and innovative Web applications"[7]

The change from traditional desktop software to web-based software is still quite recent. In the mid 1990s the Internet was used for mostly accessing static documents and pages. Early web-based applications were developed using scripting languages that were scattered and unstructured. Over a decade web engineering grew rapidly, borrowing techniques from software engineering. Initially the web was a typical client-server architecture, as illustrated in Figure 2.4.

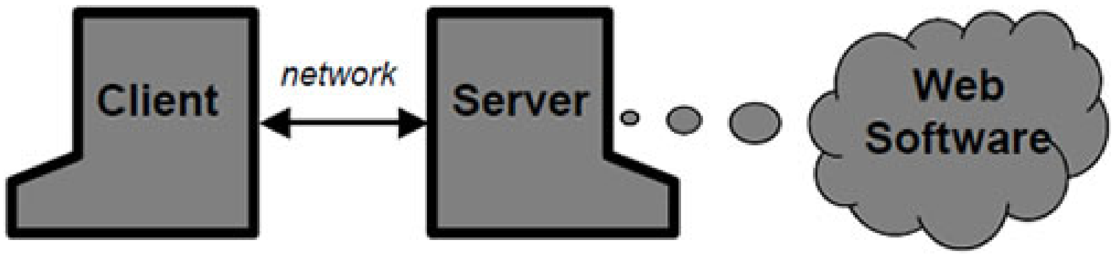

Figure 2.4 – *2-tier client-server architecture*

This simple model is sufficient for small web sites, with little functionality. The server(web-server) serves the web pages to the client(web browser) on the user's computer. Today the situation is very different[18]. This simple model does not support all the requirements of web applications. As websites evolved, the simple 2-tier client-server model became less maintainable. Today the client-server model for web-based applications tend to be n-tier. The 2-tier model of web-server and web browser still exist but the web-server now also can connect to application servers and database servers. Figure 2.5 illustrates this model.

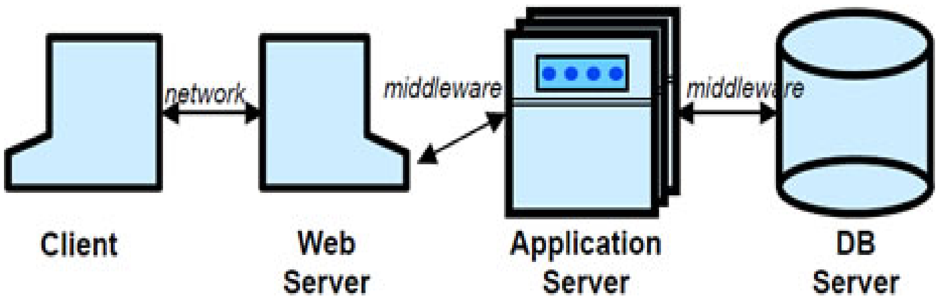

Figure 2.5 – *n-tier client-server architecture*

Application servers execute the business logic of a web application. These servers retrieve data from database servers and render a HTML document for the web-server to return to the client. The HTML document is used by the client to render the website. The HTML document can contain JavaScript and CSS scripts for additional functionality and styling. HTML forms are used by the client to send data to the web-server and to the application servers.

Advancements have not just been exclusive to the server side of the model. Web browsers have evolved to allow for a multimedia based Internet. The W3C[19] creates standards for HTML and CSS. Pressure is put on web browser developers to meet these standards. One standard shared by all web browsers allows a web browser to become a base for a universal user interface[7].

An important realisation was that the Model-View-Controller (MVC) software architecture also applied to web applications. This brought structure and better maintainability to web-based software. The MVC model, consists of three kinds of objects[20]. The model is the application object that stores,retrieves and organises data. The view renders the data for presentation. In web-based software the view renders the HTML document. The controller takes the user input and executes the business logic using the necessary view and model objects. MVC separates the programming from the presentation. This allows for greater reuse and flexibility. Figure 2.6 illustrates the abstract structure of the MVC model.

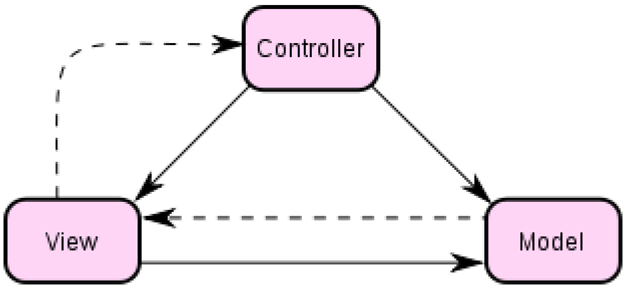

Figure 2.6 – *MVC model*

Object oriented programming plays an important part in the MVC model. Inheritance of the abstract model,view and controller objects allows a developer to define customised actions and create dynamic content views.

2.5 Web-based software versus traditional desktop based software

As previously stated there is a trend of software companies opting for web-based development over traditional desktop development. While web-based applications have limitations, the advantages allow for cost savings. There are savings in development and especially in distribution. Computers are now common household items. Home computers come pre-installed with web browser software and Internet connections are becoming faster and cheaper. This leads to users having a familiarity with the use of the Internet and web-based software. There are also additional advantages for the user. Accessing a web application allows the user to have their data remotely stored at a central location. This removes concerns about data storage and adds the convenience of allowing the user to access their data from any location.

There are limitations to web based software. Considerations have to be made for the effects of:

- No access to data due to user's Internet being disconnected.
- The burden of security is even greater on the software provider.
- Quality degradation due to slow Internet connections.

- Quality degradation due to old web browser versions.

Despite these limitations, the biggest advantage to web applications is the saving in time and installation training of deployment. The software can now be installed and updated in only one location. In the deployment of traditional desktop-based software, updates and installations had to be issued by disk or downloaded from the Internet. The developer had to check and verify the current installed version of the software before allowing the newest update to run. The developer also had no control over users running the updates. This can lead to some installations of the software becoming years out of date.

2.6 SAAS

"Software as a service model is capable of causing a sea change in the software industry"[13]. SAAS has potential to move traditional packaged desktop software to fully web-based software solutions. This removes the responsibility of installation, maintenance and upgrades from the support staff of a software company. Selling SAAS is different to selling a desktop software package. What is sold is a contract to access the software and store data rather than purchasing a software application to install. The software provider hosts both the software and the customers data in a central location, while the customer has access and support for the software for the duration of the contract. Companies that claim to produce software that fit the SAAS model usually meet a set of characteristics, including the following[13]:

Application centric. Providing access and management functionality to an application that is commercially available.

"Renting" or "Selling" application access. The customer gains access to a new application without investing in servers, support staff, application licenses and other resources. The provider can add value to the services in the application by the continuing development of that application.

Centrally Managed. The services are hosted and managed from a central location rather than individual installations on each customer's computer.

One-to-many service. The service is designed to be a one-to-many offering. The data can be accumulated and shared by a number of users rather than a single user importing and exporting data.

Delivers on the contract. The provider is responsible for delivering on the customer contract. The provider sells access to the application, so the provider must maintain and ensure access to the application.

2.7 Software Deployment in more detail

Deployment of software can be considered the final step in the software development process. While further development can continue after deployment, this process cycle still ends with a repeated deployment. Deployment covers the release, installation, activation, deactivation, update, adaptation and removal of software components(deinstallation)[21].

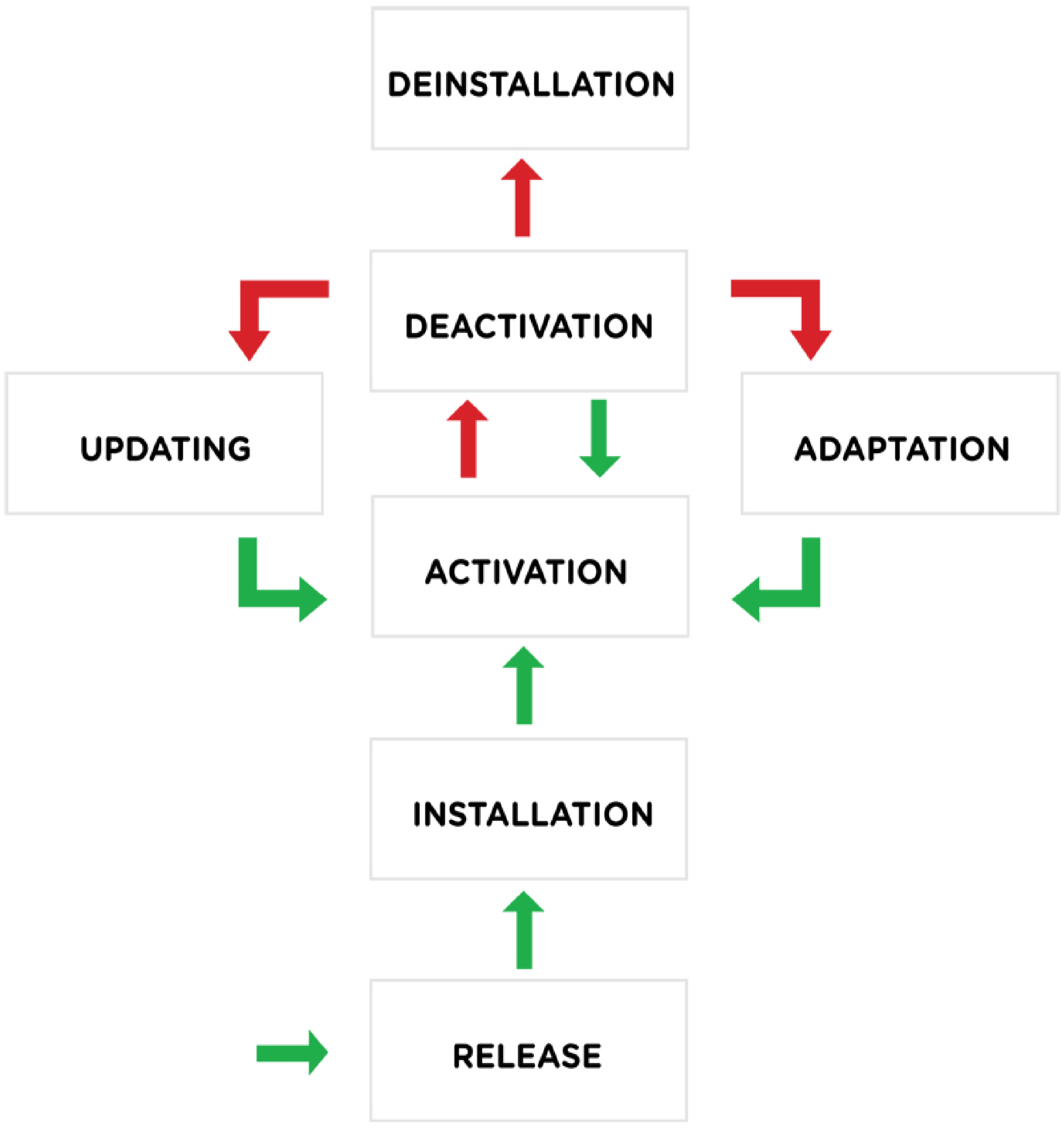

Figure 2.7 – *Deployment life cycle*

This dissertation will concentrate mostly on the installation and update processes. The deployment process is more complex than just transferring the application's files to a server. Software engineering practices encourage the reuse of components. Applications today are built on components of unique software. These reusable components provide pre-written functionality for the application.

Installation is the process of the initial insertion of a software system onto a server. It can be the most complex of the deployment tasks. Before the installation of the software system, dependencies need to be checked and installed. Once the software system files have been transferred to the server, the software needs to be configured. The software can then be activated for consumer use.

Update is the process of adapting or patching a component of the software system. Updating the software is a special case of installation[21]. During the update process, checks need to be made for any new dependencies to be installed or current dependencies to be updated. For some systems the update process may require the software system to be deactivated to block user access during the update to prevent errors occurring.

2.8 Challenges in deployment

There are many challenges and problems that need to be addressed in the deployment processes. Carzaniga describes seven problems faced during installation and update of a software system[21]. Not all of the challenges mentioned by Carzaniga are applicable to web-based system. There are some problems faced only when dealing with local desktop deployment.

Changes in hardware environment can break dependencies or induce errors in software components such as incorrect drivers for a network card.

Dependencies among components is a result of software engineering reuse practices. Dependencies increase the complexity of a deployment process. A software system on a server could break another software system sharing the hardware and dependencies if a dependency component was removed or updated.

Coordination of distributed services needs to be carefully planned. Web applications are often distributed across many servers. If an update were to change communication protocols for node 1, then node 2 needs to also be updated or reconfigured. The update process will need to either simultaneously update both node 1 and node 2 or deactivate the software system during the deployment.

Large-scale content delivery can be a burden on a system's performance. Web applications can sometimes need to process and transfer large amounts of data. Considerations need to be given to dedicated hardware for database servers or content delivery networks for large media content such as video and sound.

Managing heterogeneous platforms isn't a major issue for web applications. This shows an outdated gap in the research of software deployment for web-based software. The equivalent of this in web-based applications is that the developer must ensure that the software is compliant with all major web browser platforms.

Deployment process flexibility should meet the same standards as software engineering recommends for a software system. The deployment process should be designed so that it can be efficiently adapted to meet new and changing requirements.

Security must always be a major concern. The Internet is an open network and every server on this network is at risk of attacks. Automated deployment processes generally follow automated authentication procedures. Reliable authentication procedures must be put in place to ensure the integrity of the web application and the privacy of any sensitive data the application may store.

2.9 Approaches to deployment

"Approaches for Service Deployment" suggests there are four approaches to deploying web-based software[5].

- Manual.
- Script-based.
- Language-based.
- Model-based.

Figure 2.8 illustrates the cost in human investment against the size and complexity of the software system for the different deployment approaches.

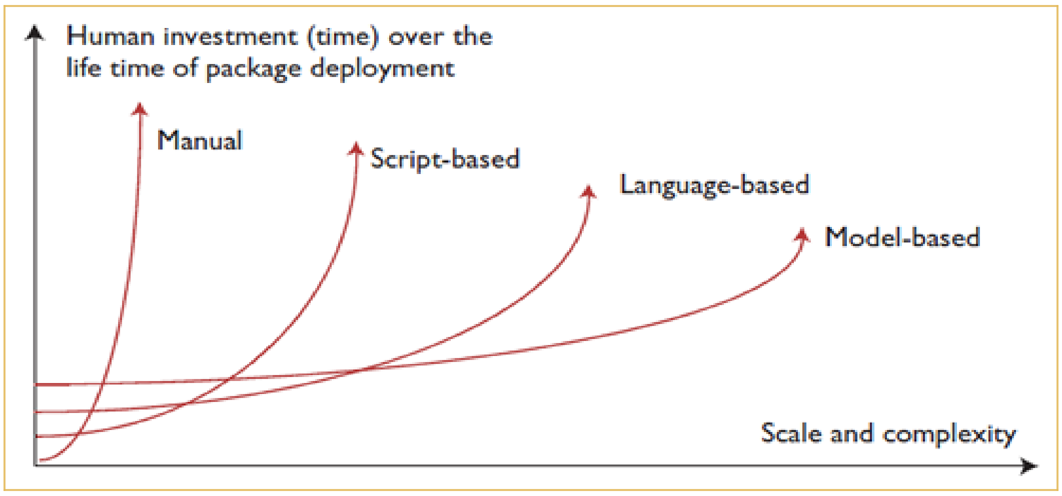

Figure 2.8 – *Human investment growth versus complexity of software system*

A manual approach to deployment can lead to many misconfigurations and inevitable deployment failure. This is caused by human error when configuring the system and installing/configuring the dependencies. A certain level of automation to suit the task of deployment reduces the time to complete deployment and also eliminates much of the human error.

Automation reduces human investment and interaction in the deployment process. Overall, this reduces costs and faults during deployment. In figure 2.8, it can be seen that increased levels of abstraction in the deployment process with higher levels of automation can in turn reduce the amount of human interaction needed. Table 2.1 below supports this statement.

Deployment Phase	**Manual Approach**	**Script-based Approach**	**Language-based Approach**	**Model-based Approach**
Development	None	Develop tools , installation and start up script templates	Develop configuration language, parser, tools, and specification templates	Develop schemas for models and tools for life cycle management, create instances of models, update software dependency model, and create resource models

Design	None	Populate application templates with customer-specific attributes and construct work flow	Populate application templates with customer specific attributes and construct work flow	Select package models from best-practice model and perform dependency analysis
Operational	Distribute packages to repository; log in to each target node; download, configure, and install; activate; and verify	Invoke distribution module, installation and ignition work flow, and verification scripts	Invoke distribution module, installation and ignition work flow, and verify notification events	Update unified interoperability model, invoke distribution module and installation and ignition work flow, and verify notification events
Change	Manually detect and adapt to changes	Discover and react to changes	Discover and react to change, and load predetermined component	Automatically react to change, reflect on model, and activate adaptation and execution

Table 2.1 – *description of deployment approaches for the deployment phases*

There are numerous tools that provide a range of facilities for deployment. There is no exhaustive list of tools for deployment and this dissertation will not attempt to provide one. The following examples of software deployment tools highlights script-based, language-based and model based respectively.

Nixies[5] is a script-based tool for deployment. It can be used to install, maintain, control and monitor applications using bash(Bourne again shell) scripts and configuration files. Nixies is suitable for small-scale systems because of the amount of configuration needed for a small system is relatively small. Nixies is unsuitable for large-scale projects as it doesn't provide an automated work flow mechanism, leading to costly human interaction.

SmartFrog[22] is a language-based tool for deployment. It is a Java-based software framework for configuring, deploying and managing distributed software systems.

Change, Configuration and Release Management from HP [23] is a model-based architecture for deployment. It is a change and configuration management solution to deploy and manage software.

An alternative example to these approaches is to use a development framework that incorporates automated deployment into it. Net-Homura[24] is an example of such a framework. Net-Homura provides PHP and JAVA APIs for the development of 3D games. These 3D games use a web-browser as the client. It packages up the developers application for manual or scripted based deployment onto a web-server.

High bandwidth content such as images and videos cause issues to the QoS if the deployment does not handle them in an efficient manner. A content delivery network (CDN) is a network of servers which host the high bandwidth content in multiple locations. When a client requests the data then a server location near to the client is chosen to deliver the content. This maximizes bandwidth access to the data and avoid bottlenecks at the main web server. The load of serving this high bandwidth content has been removed from the main web server and is now delivered in a much more efficient manner.

2.10 Relationship between deployment and quality

The cost of deployment can be reduced by ensuring a good quality of manageability(QoM)[5]. QoM is an evaluation metric of the manageability of a system. High QoM improves time to deploy and decreases the likelihood of faults caused by human error. QoM measurements include:

- Number of lines of configuration code.
- Number of steps involved in deployment.
- Time to develop, deploy and make an update.
- The ability to automate the deployment process.
- Robustness after misconfiguration.
- The ability to express constraints and dependencies.
- Configuration for first use of the deployment tool.

Automation has the highest impact of QoM. Automation reduces the amount of human interaction, which decreases human error and time to deploy. The deployment process can also affect the quality of service(QoS). The deployment process must not affect the following in a negative way:

- Reliability.
- Usability.
- Security.

- Availability.
- Scalability for high loads during peak usage times.
- Performance and low latency.

2.11 Availability of literature on deployment

There is a lack of peer reviewed publications to source information for web-based deployment. There are three primary problems:

- The majority of research and literature on deployment is for traditional desktop deployment.
- The rapid growth in web development technologies leaves much of the literature for web-based development still significant but missing important information concerning newer technologies and methodologies.
- Much of the literature on web-based deployment was found only as a paragraph or a section of a publication on software development.

The majority of the literature on deployment was found through the Association for Computing Machinery(ACM) Digital Library, the Institute of Electrical and Electronic Engineers (IEEE) Xplore, and the university library at NUI Maynooth. Medvidovic [4], Dearle [6], Jazayeri [7], and Carzaniga [21] all provide useful information and insights to deployment but the information concerning web-based deployment is limited to only a small section of the publications. Talwar [5] discusses software packages for deployment as mentioned in section 2.9. The motivation and foundations laid down by these software packages are still relevant but the evolution of web applications has left these technologies trying to catch up.

There is further information that can be found on web-based deployment but care must be taken to find reliable sources. There is much non-peer reviewed material in blog posts and tutorials created by successful web developers and web-based software companies. While this information usually cannot be verified, it provides a developer with invaluable information for a practical deployment solution. Companies like 37-signals [25] and renowned developers such as Joel Spolsky [26] regularly create blog posts sharing their experiences with software development and software deployment. Both 37-signals and Joel Spolsky produce web-based software so their publications provide information on first-hand experience of web-application development and deployment.

Another developer, Eric Ries from the company IMVU[27] writes blog entries and presents talks/lectures at software conferences on the deployment techniques used at IMVU. As part of his talks he promotes a concept called “lean startups” and “Continuous Deployment”[28]. Continuous deployment is a process that allows all code that is written for an application to be immediately deployed into production. There are arguments for and against this concept. The main advantages highlighted are:

- Immediate testing in real-world scenario.
- Frequent check-ins force developers to create modular, simple solutions.
- Optimisation of an update can be measured quickly.

It is still a controversial deployment concept that has its disadvantages and is only suitable to certain software scenarios:

- Unstructured, buggy code due to pressure time limits to meet next deployment update release.
- Requires a fully automated deployment process.
- Customers who suffer from service loss due to a bug or downtime from the deployment process can risk loss of business.

2.12 Research Summary

Software engineering is a very broad practice. During the research for this dissertation, the elements of software engineering that lay a foundation for creating a deployment process for web-based software were identified and discussed. The difference in challenges between traditional desktop deployment and web-based deployment were noted. Section 2.11 identifies a gap in research for web-based deployment and a need for more information to be collected on the practices used currently in this area. The issues discovered during this research influenced the design of the SMEs' web-based deployment survey.

Chapter 3: Software Applications and Deployment Process at Topfloor

3.1 Introduction

This dissertation uses the work placement company, Topfloor as the main detailed case study. Topfloor consists of three employees, two in development and system administration and one in support and training. Topfloor has two software systems Blockman and Letman.

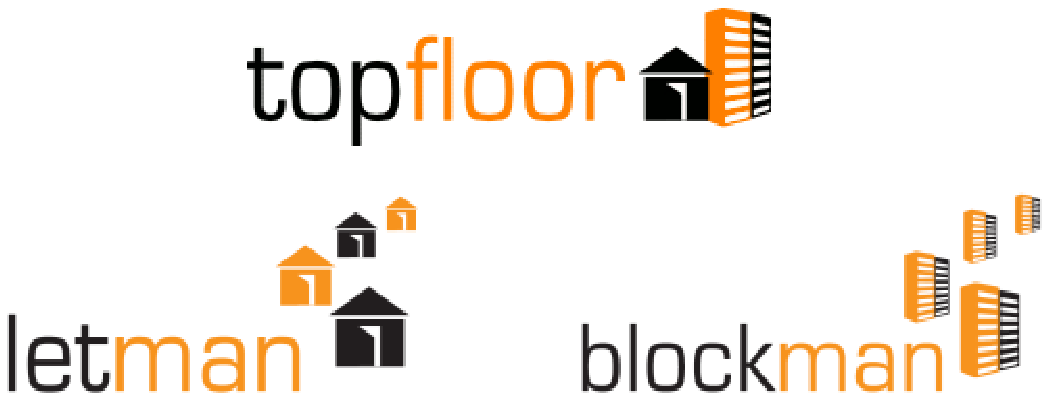

Figure 3.1 – *Topfloor software suite logos*

There are over 100 customers between Blockman and Letman. Each customer has on average five staff. This results in Blockman and Letman serving roughly 500 users.

Letman is a web based application for professional property managers(agents) involved in the letting and management of individual residential properties on behalf of landlords. Letman provides:

- Features for the advertising, letting and management of residential property.
- Features for marketing and administration tools.
- Features for managing the clients'(landlords') rent account.
- All features comply with the Residential Tenancies Act 2004.

Blockman is a web application for property managers(agents) of apartment blocks and town house and multi-unit developments with common areas which are shared amongst the property owners. It aids an agent in the running of the management company based around the property development. Blockman provides:

- Features for issuing service charge collection by debit debit.
- Accounting features such as bank reconciliation, financial year ends, audit trails.
- AGM notices / Directors Notices / Membership - Share Certificates.
- Unit details management with owner information and history.
- Creditor payment by direct credit.
- Document management.
- Call inquiry management and diary for tasks and appointments.

3.2 Hardware

Topfloor currently has three physical servers. Two are rack mountable servers. These servers contain three virtual machines each. The remaining physical server is a tower server. All servers and virtual machines have Ubuntu as the operating system. In total this is nine available servers.

One of the servers is stored locally in the Topfloor office. This server hosts subversion(SVN) for source code version control. It is also used for other internal secure information storage for the company. Developers use SVN to maintain current and historical versions of source code, web pages, and documentation. SVN allows the developers to check in their code for collaboration. It also ensures that the version of the code in each developers workspace is synchronised to the same copy.

The other two servers host the six virtual machines. These servers are stored in a large data centre. All public services, that Topfloor provide, run from these six servers. XEN is the virtualisation tool used to create and administer these virtual machines. It allows several guest operating systems to execute on the same hardware at the same time. XEN provides tools for virtual machines to be created, booted, shut down and copied. Virtualisation provides many benefits:

- Faster disaster recovery - instances of a virtual machine are just files that can be backed up, copied and restored.
- Dynamic load balancing – virtualised hardware can be easily increased/decreased depending on the required load.
- Server consolidation – a deployment setup that requires multiple servers can now be run on a single machine.

3.3 Server-side dependencies

There are a number of server side dependencies that both Blockman and Letman share. These are:

- Java JRE v1.6.
- Tomcat v6.0.
- PostgreSQL v8.4.
- ImageMagick v6.3.
- Apache FOP v1.0.
- Apache HTTP Server v2.2.

Java is an object-oriented programming language developed by Sun Microsystems. Java programs are compiled to bytecode which can only run on a Java virtual machine(JVM). The Java runtime environment(JRE) complements the JVM to run compiled Java programs. The bytecode is the same no matter what platform or operating system. There is a *"just in time"*(JIT) compiler that comes with the JVM. It translates in the bytecode into native processor instructions at run time. While Java is platform independent, the JVM is not. Only supported platforms have a JVM created for them.

Tomcat is an open source Java-based web-server for executing Java Servlet web pages. In reference to the diagram from chapter 2,

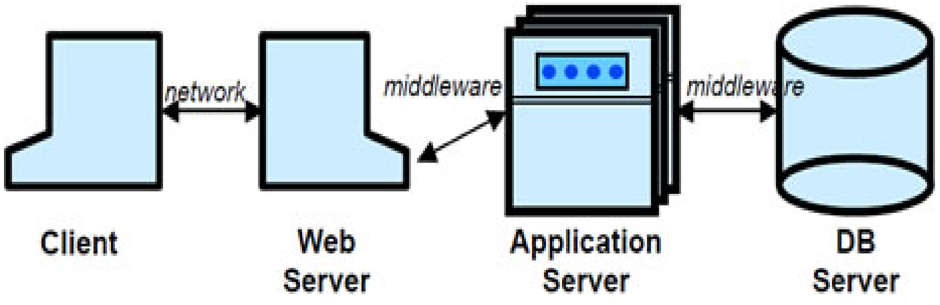

Figure 3.2 – *n-tier client-server architecture*

Tomcat is the application server. Blockman and Letman are packaged up into Jar files and loaded as libraries in Tomcat. Each client is given their own installation for Blockman and Letman. Each installation is loaded as a separate servlet in Tomcat. A servlet is a Java Object that conforms to the **Java Servlet API**, a protocol by which a Java class may respond to HTTP requests. To balance the load of incoming traffic, roughly 10-14 servlets exist in each Tomcat server. This allows for a better degrading service if one of the Tomcat servers encounters a system crash. Only 10-14 of the customers would be affected. Each Tomcat server binds to three ports. One port is listening for a shutdown signal. Once this port receives a shutdown signal, Tomcat will begin to execute the *"destroy()"* method in each of it's servlets. The second port Tomcat binds to is for HTTP requests.

Topfloor policy is to allow only local connections to this port. This is used to secure the Tomcat administrator application. The third port Tomcat binds to listens for the AJP protocol. This is the communication protocol used by the Apache HTTP server module, mod-jk. This will be discussed further in server-side dependencies. The servlet running in Tomcat makes it's own active connections to the customers database.

PostgreSQL is the database server currently used because of it's ability to handle high loads, distributed between its clusters. Other features that are very beneficial are it's ability to manage concurrent connections during SQL transactions. For extra convenience for installation, backups, updates and deinstallation, it is Topfloor's policy that each customer have their own database. For extra flexibility and efficiency there is a PostgreSQL cluster for each Tomcat server. Each PostgreSQL cluster is independent and connections are made on different ports to each cluster. A servlet uses the PostgreSQL (Java database connectivity) JDBC library to connect to it's database. Table 3.1 and table 3.2 below shows the layout example of port numbers for Blockman Tomcat and PostgreSQL dependencies:

Service	Tomcat 1	Tomcat 2	Tomcat 3	...
Shutdown Port	7001	7002	7003	...
HTTP Connector Port	8001	8002	8003	...
AJP Connector Port	9001	9002	9003	...

Table 3.1 – *sample ports bound by Tomcat servers*

Service	PostgreSQL Cluster 1	PostgreSQL Cluster 2	PostgreSQL Cluster 3	...
Listening Port	5433	5434	5435	...

Table 3.2 – *sample ports bound by PostgreSQL clusters*

ImageMagick is an open source software suite for working with images. Features include displaying, converting, and editing image files. It can read and write over 100 image file formats. Blockman and Letman use imageMagick commands to resize user uploaded images to a practical size.

FOP stands for *"formatting objects processor".* It is a Java based software suite that transforms XSL Formatting Objects(XSL-FO) documents into PDF document or some other form of printable format. It can also style these documents using XSLT styling sheets. It also allows for embedding other printable media such as images. Blockman and Letman use FOP to generate financial reports and other miscellaneous reports, as PDF for the user to download and view/print.

Apache HTTP Server is the most popular web-server software in use on the Internet. It is often referred to as just Apache. It host 54.46% of websites on the Internet while the second most popular is Microsoft IIS at 24.57% [29]. According to the same statistics, Apache HTTP serves over 112 million sites. Apache is used to serve static and dynamic content websites. Modules can be loaded into Apache to extend the core functionality. Plugin modules for example, Perl, PHP and Python allow Apache to directly execute the code. This can make Apache both a web-server and an application-server.

There are no modules with Java executing functionality suitable for Topfloor's requirements. The JK module for Apache allows Apache to forward requests and receive responses from a Tomcat using the AJP protocol. This allows Tomcat to do all the application work and rendering of HTML responses while Apache handle the web-server responsibilities of content caching, connection pooling and HTTPS encryption. HTTPS is an extension of the HTTP protocol with an encryption layer. It is Topfloor policy that all traffic in Blockman and Letman is encrypted. Apache handles encryption with the use of the SSL module and an encryption key.

Apache does one more task in the Topfloor deployment setup. It authenticates a user's connection. Each installation of Blockman and Letman is presented to a customer with a SSL certificate. This SSL certificate identifies the user. Apache will not forward the HTTP request to a Tomcat server unless this certificate is loaded and the *"Organisation Unit"*(OU) present in the certificate matches the customers URL. This will be discussed further in client-side dependencies.

3.4 Client-side dependencies

The main requirement on the client side is an adequate Internet connection. The software cannot be accessed without an Internet connection. The connection must be of sufficient speed and bandwidth, as the server may drop the session if the client is responding too slowly.

The next requirement for client-side is a modern web-browser. For Blockman and Letman to render correctly the web-browser must implement:

- HTML 4.1 or greater.
- CSS 2 or greater.
- JavaScript 1.5 or greater.

It is Topfloor policy not to support or allow connections made from Microsoft Internet Explorer. It contains too many security vulnerabilities and is mostly non-compliant with the HTML, CSS and JavaScript requirements needed for Blockman and Letman. This policy may change after Microsoft Internet Explorer 9 becomes a stable release. Topfloor support the following browsers for access to Blockman and Letman:

- Mozilla Firefox 3.0 or greater.
- Google Chrome 5.0 or greater.
- Apple Safari 3.0 or greater.
- Opera 9.0 or greater

The third dependency is a PDF document reader. Blockman and Letman produce numerous documents and export them as PDF. The PDF reader must be able to read PDFs of version 1.7 or greater. The fourth dependency is the client-side must have software to read CSV files for tasks such as mail merges.

The last dependency is the web-browser must have the authentication security SSL certificate plugged in. If the certificate is missing from the web-browser the web-server will return a HTTP 403 error.

The Blockman and Letman software can be accessed on any hardware and operating system, once the web-browser and PDF requirements are met. This includes mobile technologies such at Microsoft Windows 7 phone or the Apple iphone or ipad.

3.5 Software Design

The focus of this dissertation is software deployment, therefore the discussion of software design for Blockman and Letman is limited to a brief overview. Blockman and Letman are developed using the same MVC modelled framework. All requests start at the same point, in an object called *Dispatcher*. *Dispatcher* controls which *Controller* objects are executed. Figure 3.3 below shows a brief overview of the object structure and execution path.

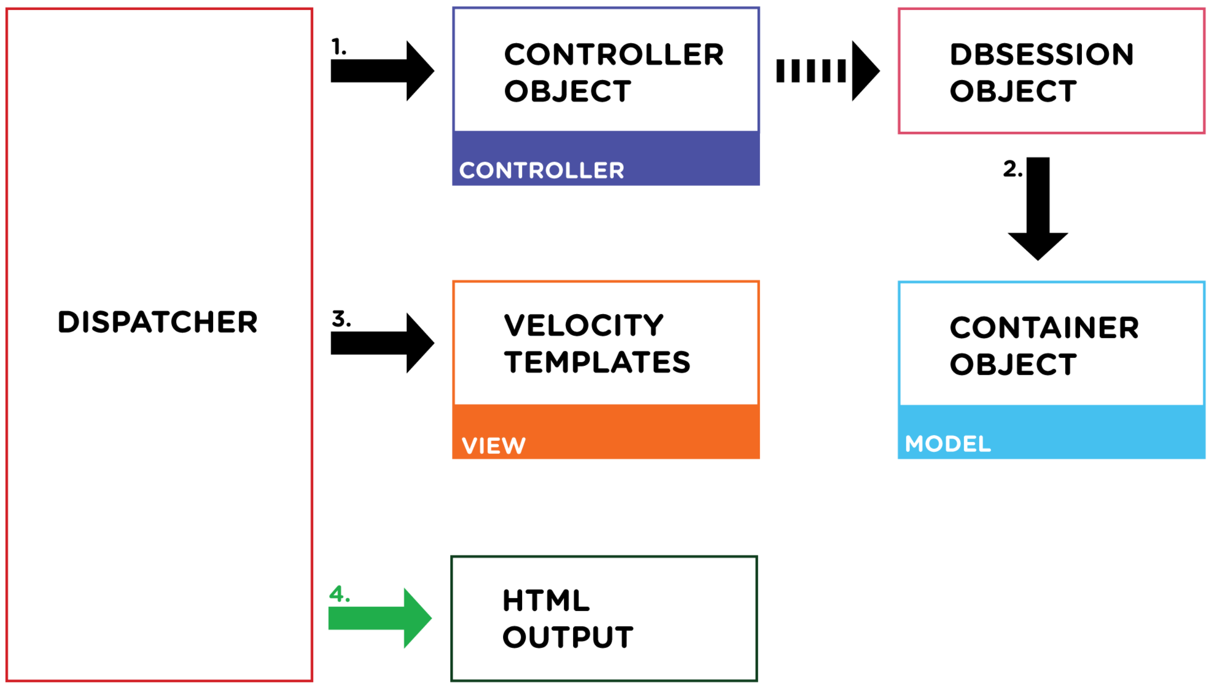

Figure 3.3 – *execution path in Blockman and Letman MVC model*

Each controller object performs a different task in the system. *Dispatcher* initialises and executes a *Controller* object corresponding to a URL parameter: *"action".* The *Dispatcher* will reject URLs that are not in the format of:

https://**domain_name**/**client_name**/dispatcher?action=**controller_name**

The *Controller* parses further user input from forms and makes the read/writes to the database. Both Blockman and Letman use a database interfacing library called Hibernate. Hibernate is an object mapping framework for Java. It maps objects directly to a database table. Each row in the database table is an attribute of the object. Figure 3.4 below shows a mapping between a database table and a object class diagram.

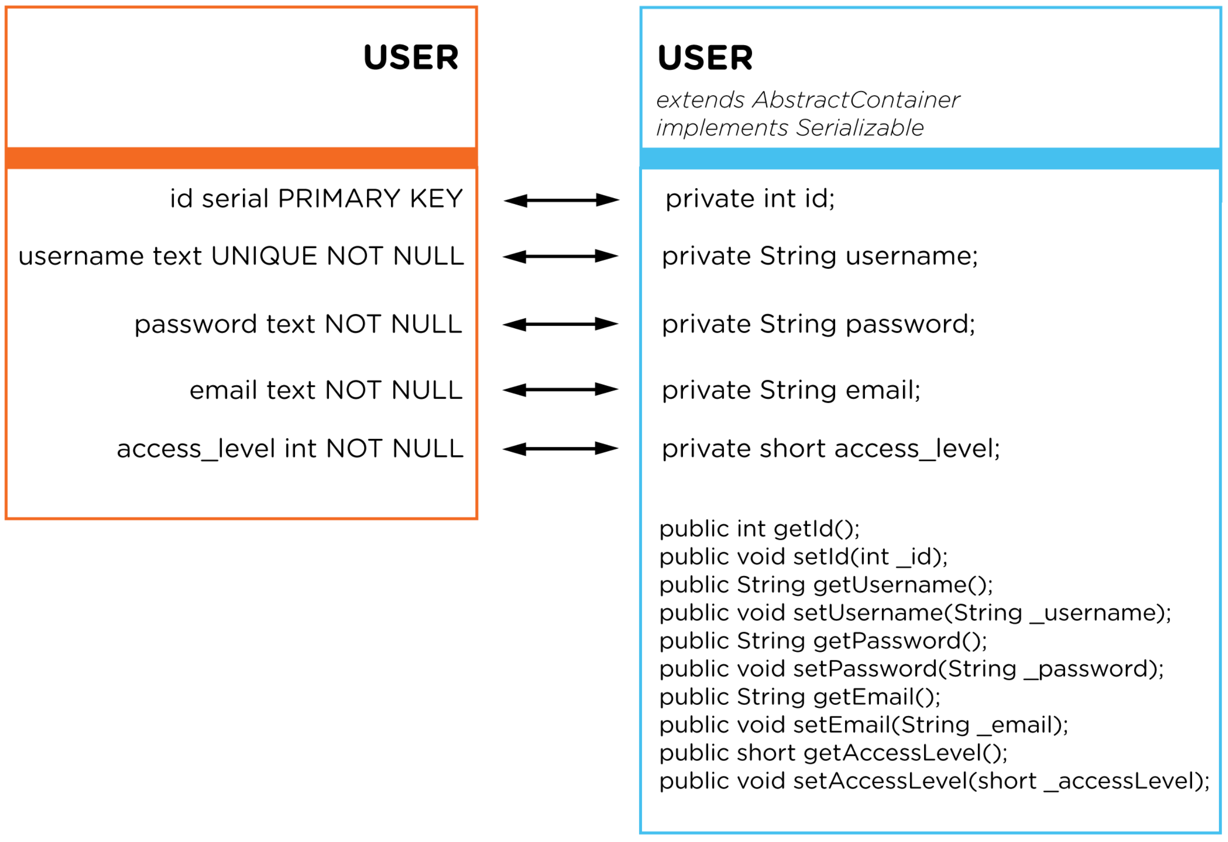
USER
id serial PRIMARY KEY
username text UNIQUE NOT NULL
password text NOT NULL
email text NOT NULL
access_level int NOT NULL
USER
extends AbstractContainer
implements Serializable
private int id;
private String username;
private String password;
private String email;
private short access_level;
public int getId();
public void setId(int _id);
public String getUsername();
public void setUsername(String _username);
public String getPassword();
public void setPassword(String _password);
public String getEmail();
public void setEmail(String _email);
public short getAccessLevel();
public void setAccessLevel(short _accessLevel);

Figure 3.4 – *Mapping between PostgreSQL user table and Java User object*

Hibernate mappings are configured using XML. Figure 3.5 shows a sample XML configuration for the User table shown in Figure 3.4.

```
<class name=”ie.topfloor.User” table=”user”>
    <id name=”id” column=”id” type=”integer”>
          <generator class=”assigned”/>
    </id>
    <property name=”username” column=”username” type=”string”/>
    <property name=”password” column=”password” type=”string”/>
    <property name=”email” column=”email” type=”string”/>
    <property name=”accessLevel” column=”access_level” type=”short”/>
</class>
```

Figure 3.5 – *XML configuration for mapping PostgreSQL table to Java Object*

Hibernate can load rows of data from a PostgreSQL table into a list of Java objects. Changes made to any of these objects can then be committed back to the database through Hibernate. To achieve this in Hibernate, the object must implement the *Serializable* interface. Altering a loaded object and then committing it, emulates the *UPDATE* action in SQL. Creating a new object and then committing it, emulates the *INSERT* action in SQL.

Once the *Controller* object has completed its necessary tasks and commits to Hibernate, it specifies a velocity template and returns control to *Dispatcher*. The velocity templates are the view section of MVC. With the use of the Apache Velocity library, the velocity templates are used to render a HTML response. The *Controller* object creates a container of variables that are used for insertion into the template file. This allows the template to provide a set layout with dynamically generated documents.

This briefly describes the MVC model of Blockman and Letman. Other libraries used in the back-end of the systems include:

- Apache log4j – for custom specified logs.
- FlexJSON – for generating JSON strings to pass data to javascripts.
- Dom4J – for parsing XML data.
- Apache Commons Fileupload – for parsing uploaded files from HTTP POST data.
- Javamail – for sending emails from Blockman and Letman software through a mail server.

The front end of the system is heavily JavaScript generated. All JavaScript code is written to use the jQuery library. *"jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development"*[30]. There is only one direct HTTP request made by the browser at login and all requests afterwards are made through AJAX calls. This is done trivially with the jQuery library. jQuery provides functionality for easy HTML, DOM and CSS manipulations and it also has many built in features for building special effects and animations. It is growing to provide simple animations that previously were easier done with Flash. The aim during software design in Topfloor was to use jQuery to create a user interface that felt like a native desktop application. AJAX allows for smaller responses from the server and as a result there is lower latency for the user.

3.6 Installation solution

In Topfloor the installation part of deployment is a mixture of script-based and manual-based deployment. The scripting tasks are done by an ANT script. ANT scripts are written in XML code. Installation is a seven step process:

Step 1. For Blockman and Letman, an ANT script compiles all the Java code and packs it into a Jar file. The ANT script then creates a build directory and copies all the files necessary for an installation into that directory. This directory contains all the CSS, JavaScript and image files necessary for the front end. It also contains a Jar file for Blockman / Letman and all the necessary Java library Jar files. It also contains all the Velocity template files.

Step 2. There is one file in this new directory that needs to be manually edited and configured. This configuration file tells the directory that it is a servlet for this particular customer. It contains details of the customer identity, the name and location of the database and the level of logging needed for this installation.

Step 3. All Tomcat servers must be shutdown to close active connections to the PostgreSQL databases. This is deactivation. If this wasn't done and the deployer attempted to make changes to a database or a Tomcat directory, the server's resources would all be used to attempt this complicated task because there would still be active connections from a client's web-browser.

Step 4. Once configured the files are transferred to a Tomcat web-app directory on the server. All of the servers are Ubuntu, a Linux based operating system. The simplest way to transfer the files to the server is by using an SCP transfer. SCP is a secure file transfer over the SSH protocol. SSH is a protocol for encrypted remote access to a server.

Step 5. A database is manually created for the new installation. The name of the database must match the name specified in the configuration file of the new servlet. The database tables and initial data must be added to the database, by inputting an SQL file through the PostgreSQL command line terminal.

Step 6. Apache is then configured to recognise a new URL for the new installation. A security certificate must be created and sent to the new customer. The “organisation unit”(OU) must be specified in Apache, so Apache can recognise and authenticate the certificate.

Step 7. Once this is finished, Apache needs to be restarted and all the Tomcat servers can be started again. This is activation.

3.7 Backup solution

The backup system is a fully automated script-based solution. The backup script runs nightly, seven days a week, every day of the year. It is a seven step process.

Step 1. The system must be deactivated. As stated in the **3.6 Installation Solution**, this involves shutting down all running instances of Tomcat.

Step 2. A dump of each database is made and the dump is stored in the “backups” directory of the matching servlet. A database dump is an export of SQL code. It is a long list of SQL statements that allow the database to be re-created in its current state at the time of the dump. As the dump is plain text, it can be compressed with a high compression yield. The dump files are compressed using the Tar bz2 compression technique.

Step 3. It is Topfloor's policy to keep 30 days of database backups. This allows for roll-backs in case corruption occurring in a customer's data, that remained undiscovered for a few days. After the database is dumped and compressed, the file is time stamped. The script then searches for timestamps older than 30 days and removes those files.

Step 4. The files for each servlet are now replicated on another server. This is done using the *"rsync"* command. The rsync command looks for changes to files and directory and only transfers the changes between servers. The transfer includes everything from the installation directory along with any file uploads from the customers e.g. images and PDFs.

Step 5. A script is executed on the backup server, that loads the current database dump into a backup database. This backup server is prepared to be an exact copy of the state of the live server. This allows for the backup server to become the main services server in case of a hardware failure in the main server.

Step 6. A second rsync is made to a server in a different location. This server is only available for file storage. This is a Topfloor policy to keep an off-site backup, as the live server and backup server as hosted in the same location. In a catastrophe scenario, the user's data is safe.

Step 7. The system is re-activated.

Figure 3.6 shows the layout of servers for the backup solution.

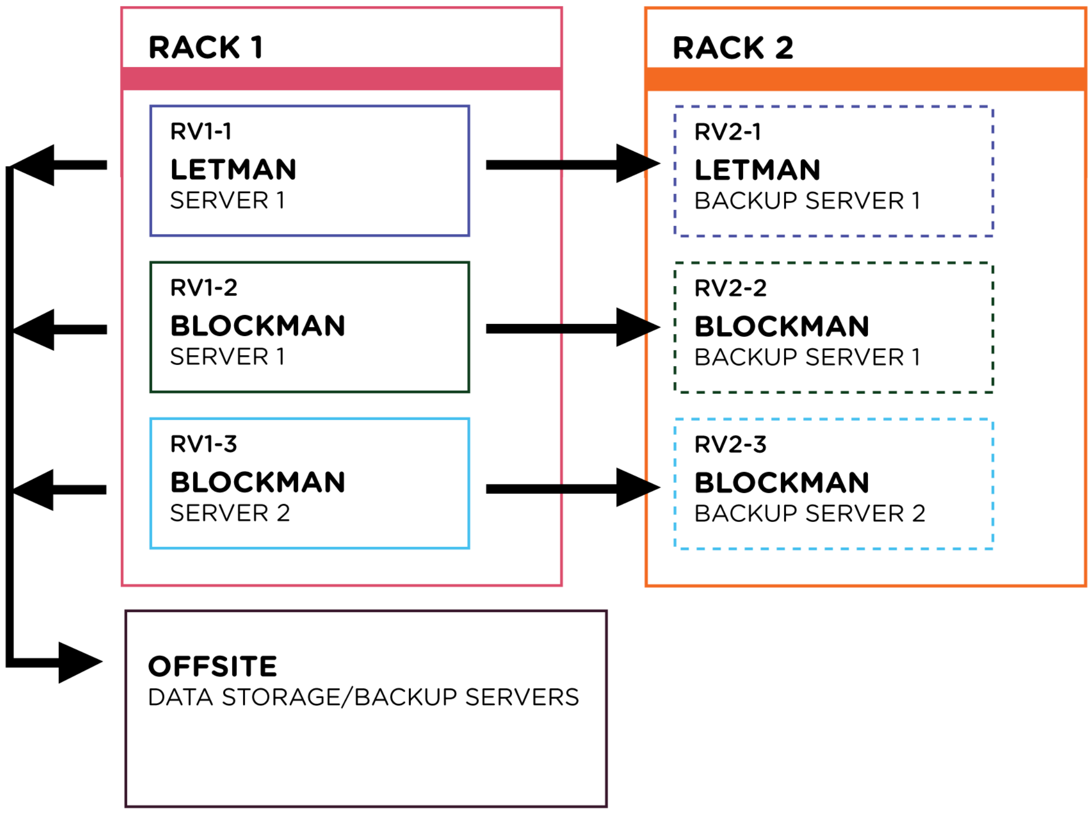

Figure 3.6 – *Overview of Topfloor's backup plan*

Rack 1 and Rack 2 are servers running XEN, located in a data centre. The RV1-1, RV1-2, RV1-3, RV2-1, RV2-2 and RV2-3 servers are virtual machines. All virtual machines on Rack 2 are mirroring data from the virtual machines on rack1. The off site backup server is hosted in a remote location seperate to Rack 1 and Rack 2.

3.8 Update solution

The update deployment process is a simpler version of installation. It is Topfloor's policy to issue updates (if an update is ready) every Sunday. If a major bug is found a fix will be pushed out as soon as the update is ready. This allows for the least amount of disruption to the user as Sunday has the lowest amount of usage traffic on the software systems. The update process is part script-based and part manual-based deployment. Update is a 5 or 6 step process depending on the kind of update involved.

Step 1. Deactivate the system.

Step 2. Run the ANT update script. This script builds a deployment directory excluding the configuration files. The configuration is already done in the installation deployment. Only the updated compiled JAR, javascripts, CSS and template files are needed.

Step 3. Transfer the files to the server with an SCP command.

Step 4. Replace the compiled JAR, JavaScripts, CSS and template files in each servlet with the new updated files.

Step 5(Optional). This only applies if there is a database patch. The patch usually consists of 2-5 lines of SQL UPDATE/ALTER commands or a CREATE table command. These lines need to be executed on each individual database.

Step 6. The system is re-activated.

3.9 Deinstallation solution

deinstallation is the simplest of all the deployment tasks. It is a fully manual-based deployment task. It is a 5 step process.

Step 1. Deactivate the system.

Step 2. Delete the servlet directory from the Tomcat instance. This will prevent Tomcat loading the servlet in the next activation.

Step 3. Delete the configuration for this servlet from Apache and revoke the SSL security certificate. The customer now does not even have access to get through Apache.

Step 4. Delete the database from PostgreSQL.

Step 5. The system is re-activated.

The removal of the servlet from the live server, will remove all the customers files from all backup servers during the next backup. The rsync command automates that part of the deinstallation process. It is Topfloor policy to give the customer a copy of their data before the deinstallation process.

3.10 Summary of Topfloor's Deployment

Topfloor's deployment system was designed through a process of trial and error and also from information found in online blogs and other internet sources. With further research and study, improvement can be made to improve efficiency and reliability of Topfloor's software systems. Documenting the entire deployment process also influenced the decisions made during the design of the SMEs' web-based deployment survey. The survey needs to obtain information that is comparable to Topfloor's size and software model.

Chapter 4: Analysis of Web-Based Survey

4.1 Introduction

For this dissertation, data was needed on how other SMEs deploy their software systems. With this data, it was planned to make comparisons and evaluations with the deployment process used in Topfloor. To acquire the data a survey was designed. The questions were designed to obtain an overview of other SMEs deployment processes. The design of the survey also took into consideration companies natural unwillingness to provide in-depth details about internal practices. The companies that participated in the survey did so anonymously and the data gathered was collated and reported in an aggregated fashion. The survey was distributed using Google Documents web form facility.

4.2 Survey Motivation

There were 14 questions in total. They were designed to investigate four key areas that influence the foundation and structure of a companies deployment process.

These three questions were included to get an idea of the size of the company.

- How many staff work on development and support?
- What is the average number of users per installation of your software?
- How many servers is your software deployed on?

These three questions were included to get an overview of the companies software hosting infrastructure.

- How is the software hosted (server infrastructure)?
- What are the operating systems used on your server(s)?
- Why was the above operating system(s) chosen?

These four questions obtain information on the areas of the development process that significantly affect the deployment process.

- What programming languages are used in the software?
- What database servers are used?
- Does your software use a third party content delivery network(CDN) for images / videos / content?
- What version control software do you use?

These four questions provide information on the deployment processes in place without forcing the survey participant to give specific details.

- How often do you issue updates?
- Do you have an automated process for issuing updates?
- Do you have an automated process for installation?
- Do you have an automated process for deinstallation?

4.3 Survey Assumptions

It was expected that there would be a limited response to the survey because of company policy to keep the deployment process confidential. While designing the survey, four key areas were prepared containing assumptions on the results. Each of the key areas detailed in section 4.2, were designed with multiple assumptions and predictions.

Key Area 1

Topfloor is an SME with three employees working on development and support. As an SME, the scale and size of the software, the number of staff and the customer base should be comparable to other SMEs. Similar to Topfloor it is expected the number of development and support staff will be 1 to 5 people. With a small customer base, staff can have multiple responsibilities, working in development, deployment and support. The software can be updated and deployed cheaply as staff are familiar with most aspects of both the software and the deployment process.

Key Area 2

A common factor in SMEs is a small budget for hardware and hosting infrastructure. It is expected that most of the companies would choose free or low budget options. For example:

- Free operating systems like Ubuntu or Debian.
- Choosing virtual servers over physical hardware servers.

Alternatively it is also expected some SMEs may use proprietary operating systems for outlying reasons. Examples for choosing such an infrastructure might be due to legacy systems or internal policies issues. The cost of re-training and migrating to a non-proprietary infrastructure could out-weight the cost of using a proprietary infrastructure such as a Microsoft Windows Server.

Key Area 3

Similar to the assumptions and predictions for key area 2 it is expected most of the SMEs would choose free or low budget options for the development processes that affect the deployment process. For example

- Non-proprietary development frameworks and programming languages such as Java or PHP.
- Non-proprietary supporting server software such as MySQL, PostgreSQL.
- Make use of a content delivery network (CDN) to reduce hosting requirements and improve end-user QoS.
- The use of version control software to manage collaboration and deployment builds such as SVN or GIT.

It is expected that the distribution range of the programming languages can be predicted. There is a vast range of scripting and programming languages for web-application development. It is predicted though that the most popular languages would be non-proprietary.

In relation to the assumptions and predictions for key area 2, there will be some SMEs that choose proprietary operating systems. These operating systems usually also have proprietary development frameworks and supporting servers packaged together (Database Server, Application Server etc.). Using Microsoft Windows Server as an example; it is expected that this operating system will be used in conjunction with a .NET framework (e.g. ASP.net) and the Microsoft SQL server.

Key Area 4

In an SME it is the expected the scale of the software system will be relatively small. The small scale of the software should allow for updates to be issued quite regularly as the system would be less complex than a large scale software system. It is also expected that the majority of the SMEs will not have a automatic deployment process for update, installation and deinstallation. The small scale aspect of the software may make the implementation of an automatic deployment process an inefficient use of time.

4.4 Results

There were 13 responses to the survey. There was one outlier response, where the data came from a company much larger than an SME. This outlier was excluded from the published results of the survey. All other responses were from Irish based SMEs that create web-based applications using the software as a service(SAAS) model. The results are discussed in this section and are divided into the four key areas specified in sections 4.2 and 4.3.

Key Area 1

The results showed that most of the SMEs were of a similar size to Topfloor. The survey indicates most of the SMES had 1 to 5 staff on development and support of the software system.

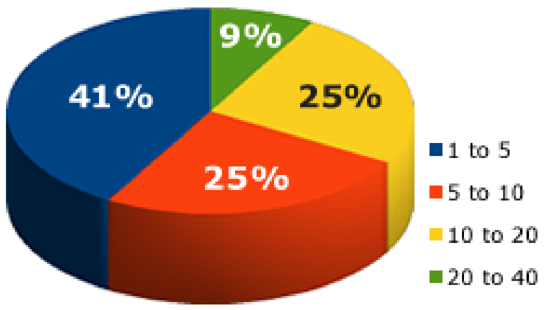

Figure 4.1 *– How many staff work on development and support?*

Similarly the survey showed most of the SMEs maintained 1 to 5 servers to distribute their software system.

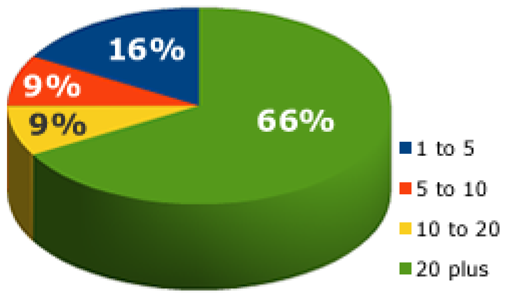

Figure 4.2 – *What is the average number of users per installation of your software?*

The majority of the SMEs showed they had 20 plus users per installation of the software.

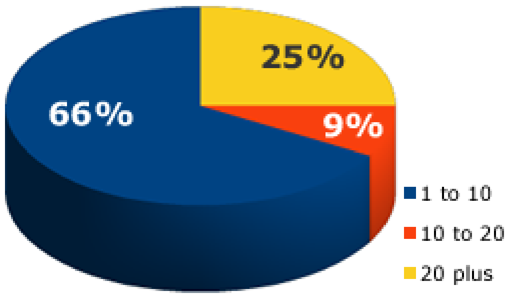

Figure 4.3 – *How many servers is your software deployed on?*

Key Area 2

The hosting infrastructure for the majority of the SMEs was also similar to Topfloor's. Virtualisation was the most popular approach to server infrastructure.

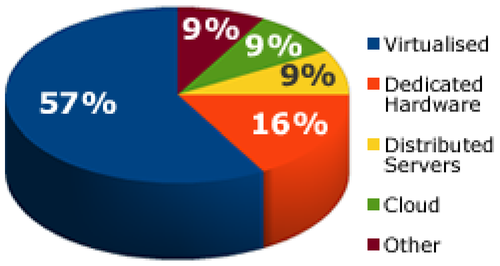

Figure 4.4 – *How is the software hosted(server infrastructure)?*

The majority of the SMEs also selected Linux / Unix as the operating system used on their servers.

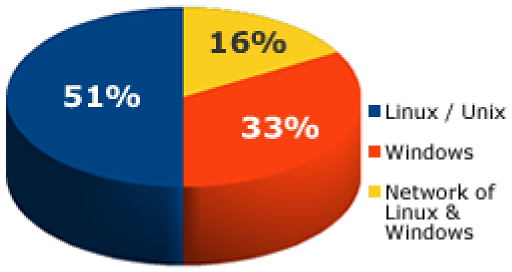

Figure 4.5 – *What are the operating systems used on your server(s)?*

The majority of the SMEs recognised open source operating systems in conjunction with virtualisation as a method of hosting their software with low costs and low maintenance. The main reason for using a non-proprietary operating system was security. The SMEs that used Microsoft Windows Server as their operating system suggested that it was their own customers' lack of trust in open source software that influenced their decision to use proprietary operating systems. Other reasons given for Windows Server was internal policies to use Microsoft software and the training courses available to staff were limited to Microsoft products.

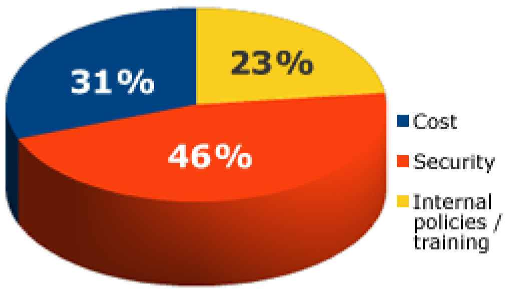

Figure 4.6 – *Why was the above operating system(s) chosen? (Multiple selection)*

Key Area 3

There was a clear distribution under the categories of proprietary and non-proprietary in programming languages chosen. PHP and Java are listed as the most popular languages among the SMEs. ASP.net from Microsoft also appeared high in the list among the SMEs using Microsoft Windows Server.

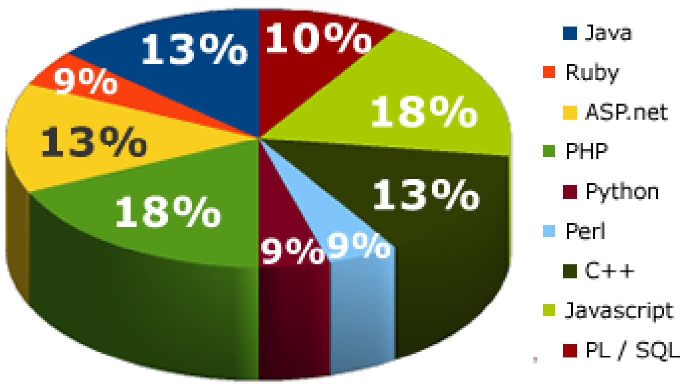

Figure 4.7 – *What programming languages are used in the software? (Multiple selection)*

There was a similar distribution between proprietary and non-proprietary systems in the usage of database servers. MySQL and PostgreSQL are listed as the most popular database servers, while Oracle and Microsoft SQL server followed as the next most selected.

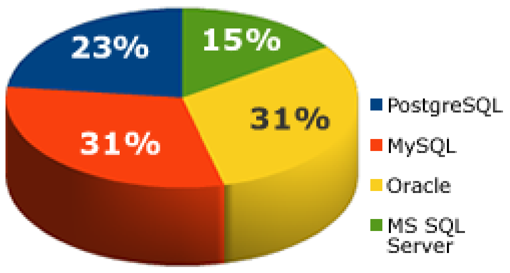

Figure 4.8 *– What database servers are used? (Multiple selection)*

The majority of the SMEs indicated that they did not use a CDN along with the deployment of their application.

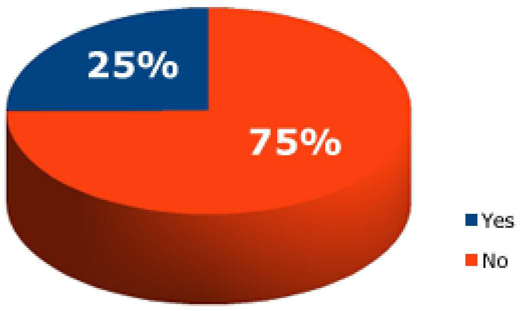

Figure 4.9 *– Does your software use a third party content delivery network for images / videos / content?*

The most popular version control software package was SVN. SVN is also used in Topfloor for source code version control, collaboration control and for bundling the software for deployment(deployment build). There was also a significant number of SMEs that indicated GIT as their version control software.

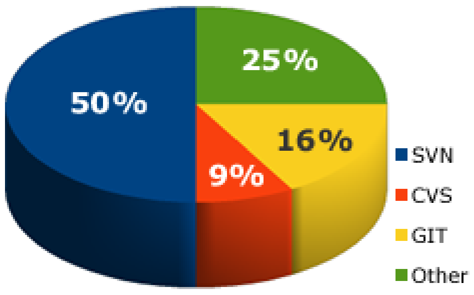

Figure 4.10 *– What version control software do you use?*

Key Area 4

The majority of the SMEs distributed updates for their software on a tri-monthly basis. It must also be noted that all of the SMEs indicated if the update contained a security update or a critical bug fix, it would be updated immediately.

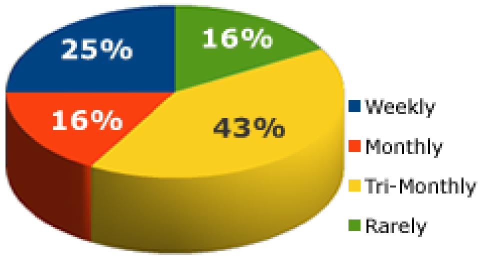

Figure 4.11 *– How often do you issue updates?*

Exactly half of the SMEs had an automated deployment process for issuing updates. The majority of the SMEs had automated deployment processes for the installation deployment. Alternatively the majority of the SMEs had no automated process for deinstallation.

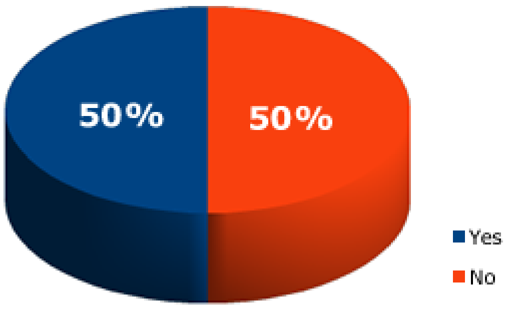

Figure 4.12 *– Do you have an automated process for issuing updates?*

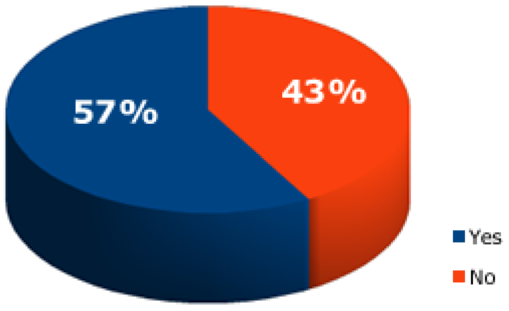

Figure 4.13 *– Do you have an automated process for installation?*

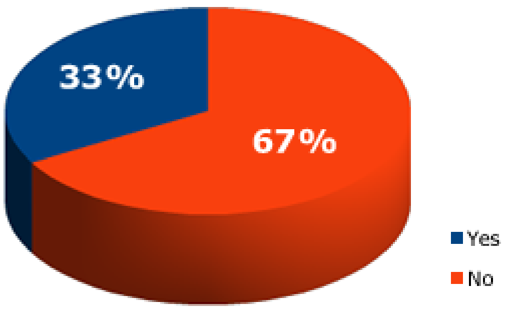

***Figure 4.14** – Do you have an automated process for deinstallation?*

4.5 Survey Analysis

Overall the survey produced a useful set of statistics. While the survey could have been extended to obtain more detailed data, the data range meets the requirements for this dissertation. The survey was designed to be short and simple to complete. It succeeded in obtaining an overview of the deployment process in a sample of SMEs. This data is very useful given that no comparable data can be found in the literature. The majority of the survey results confirmed the predictions and assumptions. However some responses provided unexpected results. The results are reviewed in detail in Chapter 5.

Chapter 5: Critique

5.1 Introduction

This chapter contains a discussion and critical analysis of the predictions and results for the four key areas of the survey from Chapter 4. In addition consideration is given to the background reading described in Chapter 2 and to the Topfloor deployment process as discussed in Chapter 3. The results of the critical analysis are summarised into five determinations that are relevant to Topfloor and other SMEs.

5.2 Key Area 1

The survey showed that the majority of SMEs had 1 to 5 staff working in development and support. This also shows that Topfloor fits into this SME group. Having a low number of staff has advantages. It's allows the company appear more consistent, focused and personal to its customer base as they can build relationships with the team and rely on known staff to reply to support issues all the time. Costs are also more manageable as staff can be responsible for multiple areas of the running of the company. This cost efficiency adds a competitive advantage.

The survey showed the majority of SMEs managed 1 to 5 servers. This corresponds with the majority of SMEs having 1 to 5 staff on development and support. A larger number of servers might require dedicated staff to manage the network.

There was one area that Topfloor differed from the other SMEs. The majority of SMEs indicated they had 20 plus users per installation while Topfloor has an average of five users per installation. This difference can be attributed to Topfloor's niche market. The customers of Topfloor are property managers who usually require only a low number of staff.

5.3 Key Area 2

The survey showed the expected result that the majority of SMEs would opt for the combination of Virtualised server and a Linux / Unix operating system. The reasons given for this matched

Topfloor's policies for using open sourced operating systems. This combination was considered the lowest budget, lowest maintenance and most secure hosting infrastructure. Virtualisation allows for cost savings in hardware and also a low maintenance cost for up-scaling; the server image just needs to be moved to larger hardware.

There was an unexpected large response of *"security"* being a major reason when choosing between Linux / Unix and Microsoft Windows. A speculated explanation for this choice is the involvement of open source communities with their own projects. Open source communities usually use the software they develop. As a result bugs and security vulnerabilities are reported and fixed faster than in a large proprietary software company. There was also the concern that a proprietary system costs more in time and money to keep secure. Security of data is a major concern when running a SAAS modelled software system. Significant amounts of the customer's data is hosted by the provider. It is the providers responsibility to ensure the integrity and security of this data.

Another unexpected result from this key area is that approximately 25% of the SMEs used dedicated hardware servers. It was expected this percentage would be smaller. For SMEs it should be more cost efficient to use virtualised hosting infrastructure. Speculated reason for their use of dedicated hardware might be:

- A lack of trust or understanding of virtualisation.
- The software system is too resource heavy to distribute on a cost efficient virtual machine.
- Some Microsoft Windows Server setups run more efficiently on dedicated hardware.

5.4 Key Area 3

As expected the majority of the SMEs chose non-proprietary development frameworks and database servers. This follows the trend that can be seen in key area 2.

A difficult statistic to predict was the usage of a content delivery network (CDN). If an SME had a significant amount of images or high bandwidth media content they would benefit cost wise from a CDN. Using a CDN would lighten the load on a server and even reduce the requirements a server needs to meet. The majority of the SMEs indicated they did not use a CDN. This may be because:

- Lack of knowledge of the availability of such a service.

- Lack of knowledge of the benefits that a CDN could add to their web-application.
- The level of high bandwidth content served by their web-application may not be significant enough to justify the use of a CDN.

Topfloor currently does not use a CDN. The software packages Letman and Blockman have the potential to store a large number of images. This dissertation suggests that response times from the servers would improve with the use of a CDN.

5.5 Key Area 4

This key area had a small number of incorrect predictions. It was discovered that the majority of the SMEs had an automated process for installation and update. Topfloor only has a small portion of the installation process automated. This dissertation will recommend Topfloor also produce fully automated processes for installation and update. All background research and survey results suggest that an automated deployment is a common step taken in creating a deployment process. It was also discovered that the benefits of an automated deployment process out-weight the cost of setting one up.

The period of time between updates also differs from Topfloor. Topfloor has regular weekly updates while the majority of the SMEs deployed updates on a tri-monthly basis. This difference with Topfloor could be accounted for again in the niche market. Topfloor produces software for a market where the demands and requirements change regularly.

The majority of the SMEs including Topfloor do not have an automated process for deinstallation. In Topfloor it was deemed an unnecessary use of time because it is a very rare occasion when an installation needs to be removed. This is also probable for the other SMEs that indicated that they did not have an automatic deinstallation process.

5.6 Determinations

The assumptions for key areas 1 to 3 were mostly accurate predictions. Key area 4 had three incorrect predictions and one correct prediction. Key area 4 also produced the most valuable insight into the deployment practices in SMEs. Using the information learnt from the survey, in combination with the research from Chapter 2, a critique can be made of the weaknesses in Topfloor's deployment process along with some recommendations for improvements.

Determination 1

Topfloor issues regular weekly updates. These updates are manually distributed. This leaves room for human error. A single installation could be overlooked or an update could be distributed incorrectly. There is also the cost of staff work hours. An automated process for update would greatly improve the QoM and prevent the issues previously mentioned.

Determination 2

While an installation deployment is not as regular an occurrence as an update deployment, nevertheless an automated installation process would be of benefit to Topfloor. There would be an improvement to QoM, where time factor cost is negligible and the introduction of bugs due to human error is eliminated.

Determination 3

An additional benefit gained from determinations 1 and 2 relates to the software system's QoS. There will no longer be any denial of service from bugs introduced due to human error. There is also the reduced service interruption time (downtime), during new installations and updates. An automated process can complete the task and re-activate the services faster than a human.

Determination 4

Moving the database servers(PostgreSQL) to a dedicated virtual server would improve the software system's QoS. It would result in a distributed software system. While it would affect the QoM in a negative manner in terms of creating addition maintenance duties, it would improve the response times from the web-server. Overall the customer's service would be improved with faster response times.

Determination 5

The final recommendation is introducing the use of a CDN. This will reduce the server and storage requirements on the hosting infrastructure. Responses times from the web-application will improve and subsequently improve the web-application's QoS. The deployment process will gain additional complexity but the impact of this additional complexity should be minimal if the deployment processes are automated.

5.7 Response to Determinations

The recommendations made in 5.6 were brought to the attention of Topfloor management and developers for discussion. Below is a summary of their professional reviews of the 5 determinations.

Determination 1

There was a positive response to the recommendation of an automated system for updates. During the discussion, improvements were made to the recommendation. The addition of allocating an exact day and time of the week for updates was proposed. This would bypass the need for the deployer to even initiate the update deployment process. A queue of updates could be made during the week. Then on the allocated day and time the update process will automatically run, checking and deploying the updates in the queue. A proposed allocated day and time was at Monday 1am. This time would cause least disruption to the customers as there is rarely someone using the software at that time.

Determination 2

There was a similar positive response to the recommendation of an automated process for installation. However it was brought up that there would be need for the deployer to manually initiate the installation deployment process. A new installation might happen at any time of the week. It was also pointed out that it may be difficult to automate the load balancing decisions. Currently the balancing of installations among the Tomcat servers is based on loose measurements and on the deployer's discretion.

Determination 3

It was agreed that the points made in determination 3 strengthened the argument for automated processes for update and installation.

Determination 4

The recommendation of altering the database server setup to a distributed network setup was met with scepticism. It was agreed that theoretically such a setup should provide improved response times and an improved QoS. However it was argued that the broadband infrastructure in Ireland, which Topfloor's customers use, was not fast enough and would negate the improvement in response times from the web-application. It was proposed that further testing be carried out on this recommendation before a decision was made.

Determination 5

It was also agreed that theoretically a CDN could provide improvements to response times and QoS. However the same arguments were made against it as in determination 4. The speed of broadband used by Topfloor's customers would negate any improvements. It was proposed that further testing be carried out on this recommendation before a decision was made.

Chapter 6: Conclusion

6.1 Dissertation Summary

During this project, gaps in empirical research for web-based deployment were identified. It was found that it was difficult to determine the best deployment techniques to implement for a given scenario, for example the deployment of the software systems in the work placement company, Topfloor. It was the aim of this dissertation to obtain information on the development aspects and the surrounding business environment aspects that influence the deployment of a web-application. With this information a critique of currently used web-based software deployment techniques could be provided. It was then hoped that with this knowledge a more informed decision could be made on which deployment techniques to use.

The research highlighted areas for this dissertation to concentrate on. It provided insight into the level of information needed to adequately analyse and critique a deployment process. It also indicated the level of detail required for the specification of the Topfloor software design and deployment process as a base case study. The findings from the research also influenced the target data required from the survey. The survey was a crucial element to this dissertation in collecting relevant information on deployment techniques currently used by SMEs.

The survey provided an overview of deployment processes used by SMEs in Ireland. While the information was not as detailed as the Topfloor deployment process specifications, the survey results were specific enough for a critical analysis to be made. This analysis highlighted the weaknesses in the Topfloor deployment process and allowed for a list of recommendations to be compiled.

6.2 Dissertation Goals

As described in Section 1.7, this project aimed to achieve five goals. Work to satisfy each of these goals was described throughout this dissertation and is summarised as follows.

Determine factors that influence the quality of a deployment process for web-based software.

In order to satisfy this goal, considerable background reading was carried out, including relevant research literature, online information resources e.g. articles, blogs, and textbooks. The review provided valuable information which was further strengthened by the survey implemented in this project.

Gather information of current deployment practices in use by Irish SMEs.

In order to gather information on the above a detailed survey was designed and circulated to a number of Irish SMEs that develop web-based software. The data collected provided an aggregated overview of deployment processes used by these companies.

Analyse and critique current deployment practices in use by Irish SMEs.

As outlined in Chapter 5 a detailed analysis and critique of the survey results was carried out. In this analysis, the unexpected results and findings were highlighted and several conclusions were made based on these discoveries.

Analyse Topfloor's deployment process and make recommendations.

Chapter 3 provides detailed documentation on the background to Topfloor's software and the deployment techniques used. Section 5.6 in Chapter 5 builds a list of recommendations to improve Topfloor's deployment processes and Section 5.7 provides a summary of the response from management at Topfloor to these recommendations.

Identify areas for further research into web-based deployment.

Section 6.2 identifies areas of interest for further research into web-based deployment.

6.3 Future Research

There is a need to obtain further information from other SMEs about their deployment practices for web-applications. Additional research would be of benefit to industry and also to further evolution of the internet. This could be done with an extended survey or a face-to-face interview process with the current survey participants. The goals of a continued investigation should provide accurate insight into the reasons for the unexpected results from the survey distributed as part of this dissertation. The extended information on these SMEs would also benefit the construction of a formal specification of quality attributes for web-based deployment processes. To create this list of quality attributes there is a need for complete specifications similar to the Topfloor deployment process specification shown in Chapter 3. The customer's view of the deployment cannot be ignored. Further study also needs to investigate the affects on the quality of service that a customer receives from the deployment process.

This dissertation provides the first step to into the research of the factors that determine the quality of a deployment process for web-based software. Deployment is a broad area in the practice of software engineering with many influential factors. While the data and conclusions made in this dissertation provide useful information and insights for SMEs, there is still need for a formal specification of quality attributes of web-based deployment processes. There is also the issue of weather the research and conclusions made here, can be of relevance companies larger than an SME.

Appendix

Acronyms

ACM - *Association for Computing Machinery*

AJAX – *Asynchronous JavaScript and XML*

AJP - *Apache JServ Protocol*

CDN – *Content Delivery Network*

CSS - *Cascading Style Sheets*

DNS - *Domain Name System*

HTML - *HyperText Markup Language*

HTTP - *HyperText Transfer Protocol*

HTTPS - *HyperText Transfer Protocol Secure*

IEEE - *Institute of Electrical and Electronics Engineers*

IP (address) - *Internet Protocol*

JIT – *Just in time*

JRE - *Java Runtime Environment*

JS - *JavaScript*

JVM - *Java Virtual Machine*

MVC – *Model, View, Controller*

PHP - *Hypertext Preprocessor*

QoM – Q*uality of Manageability*

QoS – *Quality of Service*

SAAS – *Software As A Service*

SME - *Small to Medium Enterprise*

SVN - *Subversion*

UML- *Unified Modelling Language*

URL - *Uniform Resource Locator*

WWW – *World Wide Web*

XHTML - Extensible *HyperText Markup Language*

XML - *Extensible Markup Language*

Diagram And Table Accreditation

[Figure 2.1] - Gamma, E. et al (1994). Design Patterns, Elements of Reusable Object-Oriented Software

[Figure 2.4] - Offutt, J. (2002). Web Software Applications Quality Attributes

[Figure 2.5] - Offutt, J. (2002). Web Software Applications Quality Attributes

[Figure 2.8] - Talwar, V. et al (2005). Approaches for Service Deployment

[Table 2.1] - Talwar, V. et al (2005). Approaches for Service Deployment

[Figure 3.1] - Topfloor (2010) Topfloor Property Management Software [Online]. Available from: http://topfloor.ie [Accessed: 1st October 2010]

[Figure 3.2] - Offutt, J. (2002). Web Software Applications Quality Attributes

References

[1] – Topfloor (2010) Topfloor Property Management Software [Online]. Available from: http://topfloor.ie [Accessed: 1st October 2010]

[2] - Deloitte (2009) *Deloitte Technology Fast 50 Award Winners 2009* [Online]. Available from: http://www.deloitte.com/view/en_IE/ie/industries/tmt/technology-fast50-awards/cae8a5dce4888210VgnVCM100000ba42f00aRCRD.htm [Accessed: 10th November 2010]

[3] - Deloitte (2010) *Deloitte Technology Fast 50 Award Winners 2010* [Online]. Available from: http://www.deloitte.com/view/en_IE/ie/industries/tmt/technology-fast50-awards/0350bc198553c210VgnVCM2000001b56f00aRCRD.htm [Accessed: 10th November 2010]

[4] – Medvidovic, N. & Malek, S. (2007). Software Deployment Architecture and Quality-of-Service in Pervasive Environments

[5] – Talwar, V. et al (2005). Approaches for Service Deployment

[6] – Dearle, A. (2007). Software Deployment, Past, Present and Future

[7] – Jazayeri, M. (2007). Some Trends in Web Application Development

[8] – IETF (1999) *Hypertext Transfer Protocol – HTTP/1.1* [Online]. Available from: http://www.ietf.org/rfc/rfc2616.txt [Accessed: 1st October 2010]

[9] – IETF (1994) *Universal Resource Identifiers in WWW* [Online]. Available from: http://www.ietf.org/rfc/rfc1630.txt [Accessed: 1st October 2010]

[10] – IETF (2000) *HTTP Over TLS – HTTPS* [Online]. Available from: http://www.rfc-editor.org/rfc/rfc2818.txt [Accessed: 1st October 2010]

[11] – W3C (2010) *HTML5* [Online]. Available from: http://dev.w3.org/html5/spec/spec.html [Accessed: 1st October 2010]

[12] – Hripcsak, G., Cimino, J. & Sengupta, S. (1999). WebCIS: Large Scale Deployment of a Web-based Clinical Information System

[13] – Software & Information Industry Association (2001). Software as a Service: Strategic Backgrounder

[14] – Glass, R. (1997). In The Beginning, Recollections of Software Pioneers

[15] – Naur, P. & Randell, B. (1968). Software Engineering: Report of a conference sponsored by the NATO Science Committee

[16] – Brooks, F. (1986). No Silver Bullet — Essence and Accident in Software Engineering

[17] CMS (2008). *Selecting a Development Approach.* [Online] March 2008. Available from: http://www.cms.gov/SystemLifecycleFramework/Downloads/SelectingDevelopmentApproach.pdf [Accessed: 10th January 2011]

[18] – Offutt, J. (2002). Web Software Applications Quality Attributes

[19] – W3C (2010) *World Wide Web Consortium* [Online]. Available from: http://www.w3.org/ [Accessed: 1st October 2010]

[20]- Gamma, E. et al (1994). Design Patterns, Elements of Reusable Object-Oriented Software

[21] – Carzaniga, A. et al (1998). A Characterization Framework for Software Deployment Technologies

[22] – SmartFrog (2009) SmartFrog [Online]. Available from: www.smartfrog.org [Accessed: 1st October 2010]

[23] – HP (2010) *Change, Configuration and Release Management* [Online]. Available from: https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11^833_4000_100__ [Accessed: 1st October 2010]

[24] Carter, C., Rhalibi, A. & Merabti, M. (2009). Homura and Net-Homura: The Creation and Web-based Deployment of Cross-Platform 3D Games

[25] – 37Signals (2010). Signal vs. Noise, a weblog by 37signals [Online] Available from: http://37signals.com/svn [Accessed: 10th January 2011]

[26] – Joel Spolsky (2010). *Joel On Software* [Online]. Available from: http://www.joelonsoftware.com/ [Accessed: 10th January 2011]

[27] – IMVU (2010) *Chat, Games & Avatars in 3D* [Online]. Available from: www.imvu.com [Accessed: 10th November 2010]

[28] – Lean Startups (2011) *Lean Startup Blog* [Online]. Available from: http://leanstartups.com [Accessed: 16th January 2011]

[29] – Netcraft (2010) *February 2010 Web Server Survey* [Online]. Available from: http://news.netcraft.com/archives/2010/02/22/february_2010_web_server_survey.html [Accessed: 10th November 2010]

[30] – jQuery (2010) *The Write Less, Do More, JavaScript Library* [Online]. Available from: http://jquery.org/ [Accessed: 10th November 2010]