



Stephen Lockerbie

## **DEVELOPING A BUSINESS INFORMATION SYSTEM**

Case: Work Practice Placement Management Tool for Oulu Vocational College, Department of Social and Health Care, Kontinkangas Unit

## **DEVELOPING A BUSINESS INFORMATION SYSTEM**

Case: Work Practice Placement Management Tool for Oulu Vocational College, Department of Social and Health Care, Kontinkangas Unit

Stephen Lockerbie  
Degree Programme in Business  
Information Technology  
Spring 2013  
Oulu University of Applied Sciences

## ABSTRACT

Oulu University of Applied Sciences  
Business Information Technology

---

Author: Stephen John Lockerbie

Title of thesis: Developing a Business Information System Case: Work Practice Placement Management Tool for Oulu Vocational College, Department of Social and Health Care, Kontinkangas Unit

Supervisor: Liisa Auer

Term and year when the thesis was submitted: Spring 2013

Number of pages: 31

---

## ABSTRACT

The purpose of this thesis is to develop a Business Information System and record its creation process. The main focus of the thesis is to develop a tool for managing information about student work placements. This thesis was commissioned by the Oulu Vocational College (OSAO) and its Department of Social and Health Care in Kontinkangas. The knowledge base of this thesis will define key concepts used in the thesis and consider current software development lifecycle frameworks. The practical part of the thesis developed an operational information system that fulfilled the requirements of the commissioner and the aims of the thesis.

Developing a new system or application can represent a major investment of time and money for an organisation. Often, such systems are driven by business process reengineering to achieve improvements in cost, time, service, and/or quality. These projects require careful planning and clear communication. Development frameworks for new software have moved away from the waterfall framework towards agile development. Software development lifecycles can be considered as an abstract representation for the process of creating software and provide an overall strategy for development and project planning.

The developed system matched expectations and offered the commissioner viable solutions to a variety of challenges. The methodology selected for developing the system proved to be well suited to the task. The changes that were required to the Agile Scrum framework made it more suitable for a single person development team, while the sprint cycles allowed the work to be divided into manageable parts.

---

### Keywords:

*Software development, student work placement, php, mysql*

## CONTENTS

1	INTRODUCTION	5
2	CURRENT SYSTEM	6
3	DEVELOPMENT METHODOLOGY	9
3.1	Software Development Lifecycles	9
3.2	Software Development Lifecycle Methodologies	13
3.2.1	Spiral Model	13
3.2.2	Agile	15
4	SYSTEM DEVELOPMENT	18
4.1	Preliminary Work	19
4.2	Product Backlog	20
4.3	Sprint 1	20
4.4	Sprint 2	22
4.5	Sprint 3	23
4.6	Sprint 4	24
5	CONCLUSION AND DISCUSSIONS	28
	REFERENCES	30

# 1 INTRODUCTION

The purpose of this thesis is to develop a Business Information System and record its creation process. A basic definition of a Business Information System (BIS) is considered as “a computer system within a company or organization for sharing information” (Cambridge University Press 2013, Date of retrieval 20.04.2013). The main focus of the thesis is to develop a tool for managing information about student work placements. The end product, in addition to this written work, is a working prototype tool for the specific requirements of the commissioner, the training of the relevant staff in the use of the said tool, the materials pertaining to the training as well as specific templates.

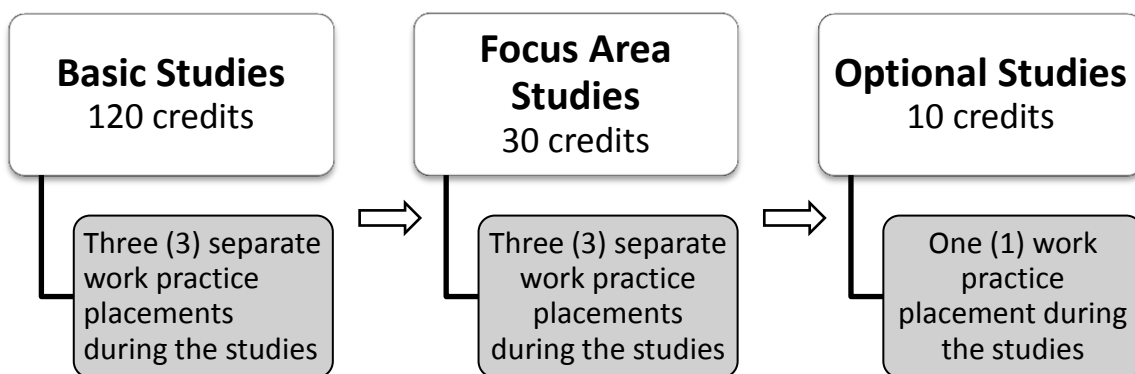
This thesis was commissioned by the Oulu Vocational College (OSAO) and its Department of Social and Health Care in Kontinkangas. OSAO is a multidisciplinary vocational institute offering training to 11300 students in 12 units in the Oulu region. Kontinkangas Unit has approximately 1150 students and 150 staff. All of the students undergo up to seven separate and different practical work placements worth 13 credits during their studies. The acquisition, administration and recording of these placements have been achieved using several different programs, methods and files on a computer. The commissioner has need of an information system to manage information regarding these work placements. Initially this system will be implemented only for the Department of Health and Social care (DHSC) in the Kontinkangas Unit. However, if the system proves to be effective, it is possible that it will be deployed in other units.

The thesis work is split into distinct parts covering both the knowledge base and practice of BIS creation. Primarily, the knowledge base of this thesis will define key concepts and consider current software development lifecycle frameworks. Current literature i.e. printed books, electronic books, and webpages will be used to define the knowledge base. The aim is to briefly define and explain some software development lifecycles. The advantages and disadvantages of each of these will be considered and one will be selected for use.

The practical part of the thesis will focus on creation of a tool for managing information about student work placements and the application of the selected software development lifecycle. The aim will be to develop an operational information system that fulfils the requirements of the commissioner and meets the aims of the thesis. It is worth noting that the work of this thesis will focus on the work of a single person development team. As such, the outcomes and conclusions may not represent a balanced view for larger teams.

## 2 CURRENT SYSTEM

A normal degree programme for the Department of Social and Health Care (DHSC) requires students to undertake seven individual work placements. The structure of these placements is illustrated in Figure 1. This process generates a large volume of data covering all aspects of work placements. All student information is stored using a database system (PRIMUS) while information regarding companies, available placements, and lecturer responsibilities are stored elsewhere. Below, this system and the processes connected will be explored. Information regarding the current system was gathered through formal and informal meetings with the commissioner and also through observing the current system in use.



*FIGURE 1. The structure of the studies and work placements at OSAO Department of Social and Health Care*

Currently, the DHSC has no formal information system for managing student work placements. All information pertaining to work placements, such as company contact details, is stored in one desktop computer and is accessible only by the Lecturer Tuula Anttila and her assistant. The information is stored across numerous Microsoft Word and Excel documents filed across numerous folders. There are a number of problems which have stemmed from this method of information management.

One of the major problems highlighted during the review of the current process was the high volume of manual work that is required to maintain and update vital information. As mentioned, student and lecturer information is currently stored on PRIMUS. Transferring this information into the local process involves a high degree of manual labour. Information is taken from PRIMUS and

then copied into the appropriate documents. Company information is manually collected and stored across a variety of documents. All of this information is manually updated, further increasing the workload.

Further to this, obtaining useful and useable information can be an arduous process. For example, some work placement providers will not accept students who are younger than 18 years old and currently the age of these students is calculated as such. First, the student group information is printed out from PRIMUS containing each student's personal identity code. This 11 digit code uses the form DDMMYYCNNNQ where DDMMYY represents the date of birth, C identifies the century of birth, NNN acts as a personal identification number, and Q is a check sum character (Wikipedia.org 2013, Date of retrieval 24.05.2013). This six digit number is then used to manually calculate the age of the student on a given date.

The 2013-14 enrolment for the DSHC estimates that there will be forty student groups each containing approximately fifteen students. This means that there are potentially 4200 manual age calculations. While observing the current system in use, it was estimated that each calculation would take approximately 1 minute. Overall, this means that the age calculation for the all of the student groups has the potential to use 70 working hours.

Information maintenance was also highlighted as a problem with the current system. As mentioned, a large portion of the information stored is manually inputted and maintained by Tuula Anttila and her assistant. With the current process, the information regarding companies is spread over a large number of documents. This means that locating information regarding companies can be time consuming. Further to this, there are challenges faced when trying to keep the contact information for companies up to date. Currently, there are over 200 partner companies that provide work placements for DHSC. Ensuring that the information is current represents a significant time investment. Even with this, it is possible for incorrect information will be overlooked. Recently, a lecturer was scheduled to visit a student in a work placement. The address information for the company was outdated. This led to inconvenience for the lecturer, student, and work place representative.

Information access is also considered to be a problem. With the current system, all information requests and change requests must be passed to Tuula Anttila for processing. This represents a significant information bottleneck. The primary cause for this is the way in which the information is stored. As mentioned earlier, almost all of the information regarding work placements is stored

within a single machine. With the current system, and process, multi-user access and editing is not possible. It is worth noting that the access restrictions are not primarily due to security issues. Although some of the information held regarding students, specifically the personal identity number, could be considered sensitive, the majority of the information can be considered mundane.

During the meetings with the thesis commissioner and with the staff working with the current process it has been possible to gain an insight into the problems presented by the current process. Primarily, the development task will aim to provide a system which will reduce the amount of manual labour required to maintain the system and improve access to the available information. The problems highlighted are considered to be the most pressing by the commissioner. As such, designing solutions to these will be the focus of the practical part of the thesis.



### **3 DEVELOPMENT METHODOLOGY**

Developing a new system or application can represent a major investment of time and money for an organisation. Often, such systems are driven by business process reengineering where an organisation wishes to achieve improvements in cost, time, service, and/or quality by improving their business process (The United States Department of Justice 2003, Date of retrieval 20.03.2013). Regardless of the final aim, these projects require careful planning and clear communication.

In the past, the majority of software was developed using the waterfall framework. This framework was heavily influenced by manufacturing practices and focuses on an uninterrupted, step by step process. One of the major disadvantages of the waterfall framework is the need to repeat all of the steps in the process if there is a change in the software requirements. As software engineering has matured the development frameworks for new software have moved away from the waterfall framework. Modern development frameworks focus more on agile development, change integration, and risk management.

The following chapter introduces and defines software development lifecycles and also explores development frameworks for use in the practical work of this thesis. The aim is to offer the reader an overview on the core concepts of software development lifecycles and to offer reasoning behind the framework selection for the thesis work.

#### **3.1 Software Development Lifecycles**

Software development lifecycles (SDLC) can be considered as an abstract representation for the process of creating software and are used to define the steps, methods, tools, and deliverables of a software project. In essence, a software development lifecycle provides an overall strategy for software development and a structure for software project planning. (Maciaszek & Liong, 2005, 15)

There are a variety of lifecycle frameworks available for software development projects. While each framework differs with regards to the importance of each phase and the particular ways that those phases, for the majority there appears to be key phases that are common to each framework. These phases can be identified as Requirements Analysis, System Design,

Implementation, Integration and Deployment, and Operation and Maintenance. (Maciaszek & Liong 2005, 15-22) (Flynn 1998, 99-130)

Requirement Analysis can be described as the activities which determine and specify the software requirements. As such, this could be considered the foundation for all further work. Maciaszek & Liong (2005, 15-16) suggest that this analysis can be subdivided into two distinct elements; determination of user requirements and requirements specification.

User requirements are a means of representing the services that software is expected to provide and provide insight regarding the software the user wants, how the user expects the software to interact with their organisation, and the benefits that the user expects to gain from the software (Flynn 1998, 128). Producing the initial user requirements can be achieved through a number of techniques. These include interviews, questionnaires, observing users interacting with the current system or process, or studying other existing software which operate within similar environments (Maciaszek & Liong 2005, 16). However, even with these techniques, forming user requirements can be challenging. Users are often unsure of what they require or what they expect the software to do. Miscommunication or misunderstanding during the creation of user requirements can lead to situations where the software does not match the needs of the customer (Maciaszek & Liong 2005, 16). With clear communication, however, it is possible to produce user requirements that represent the software or service that the end user desires.

After determining user requirements, it is then possible to specify the requirements in greater detail. The aim here is to describe the software using a mixture of diagrams, such as use case diagrams or activity diagrams, and language which is focused on descriptive explanation rather than technical details. This information can assist in understanding the complex software, as it provides a graphical overview of the operation of the software. (Maciaszek & Liong 2005, 16-17)

By combining the user requirements and requirements specification, a requirements document can be produced. This should contain diagrams and descriptions for both software services and software constraints. Software services can be divided between functional requirements, what the software does, and data requirements, the input required for the software to function. Software constraints describe the constraints that are placed upon the software. These can include legal restrictions, such as data protection and privacy laws, and requirements regarding performance and security. (Maciaszek & Liong 2005, 17)

System design offers a structured view of the software that is to be implemented, the data that will be used, the interaction and interfaces between components. While it can be argued that, in theory, there is a clear demarcation between analysis and design, in practice the division is not necessarily so clear. Maciaszek & Long suggest two reasons for this. Firstly, many modern lifecycle frameworks are incremental. As such, there may be several incrementally different versions during software development. Secondly, the modelling language used during system design will, in most cases, be the same. This means that the models created are likely to be elaboration on previous models rather than completely new. However, there is a significant difference between analysis and design. During the analysis software and hardware constraints can largely be ignored. Conversely, system design is based around the consideration of these constraints. The distinction can further be seen as the analysis models are further elaborated upon with increasingly detail specifications, giving rise to a detailed design. (Maciaszek & Liong 2005, 17-18)

Further to this, system design will normally result in the creation of an architectural plan. As the name may suggest, this plan focuses on the structure that the detailed design will follow. It outlines the structure of the software components and the way that those components will communicate. Beyond offering structure to the detailed design, the architectural plan should also lead to software that is understandable, maintainable, and scalable. Understandable should not suggest that the software is simple; instead the software and the process behind it should easily explainable to users, regardless of their technical knowledge. There are few systems that are designed to operate without maintenance. As such, the architectural design should address the need of the final system with regards to required maintenance and downtime. Scalability focuses on the ability of the software to meet growing demand from the users. Very few businesses or business systems exist without growth and this should be reflected in the architectural design. (Maciaszek & Liong 2005, 17-18)

Implementation is primarily concerned with programming the software. During this point of the SDLC, the information from the previous activities is used to produce the component parts for the software. Normally, implementation will also include debugging and testing processes to maintain software quality and functionality. With debugging, the aim is to identify and eliminate errors in the program syntax or logic structure which can be achieved using visual inspections or commercial tools. Testing, as the name implies, is seeing if the code will run. This can be achieved through a number of methods including code reviews, such as walk-throughs and peer reviews, and execution based testing (Maciaszek & Liong 2005, 18-19). Execution based testing can be

achieved in a number of methods and these can be generalised as black box, white box, and grey box testing. Black box testing means that the tester has no knowledge of the internal structure of the software being tested (Software Testing Fundamentals 2012a, Date of retrieval 15.03.2013) and the tester is normally testing specific input / expected output operations. Conversely, white box testing is undertaken when the tester has access to the internal structure of the software and test cases are built upon this knowledge (Software Testing Fundamentals 2012b, Date of retrieval 15.03.2013). Grey box testing is the middle point between black and white box testing. With grey box testing, test users have partial access to the internal structure of the software and can base test cases around this. However, tests are executed in a black box environment where the tester is unaware of the internal workings of the software (Software Testing Fundamentals 2012c, Date of retrieval 15.03.2013). The overall aim during implementation is not only to produce the software components but also to ensure functionality and quality.

Integration and Deployment represent a significant phase in software development. During integration, the various components for the software and system are brought together as a whole. It should be noted that with some SDLC frameworks integration occurs continuously throughout development and the division between integration and testing can become blurred. (Maciaszek & Liong 2005, 19)

Generally, the deployment phase of an SDLC indicates that the software and hardware of a system have been deployed alongside the normal operations of the customer. Prior to this, software has usually been tested within a local environment by the developer; this is sometimes referred to as alpha testing. This is normally followed by a period of user testing; also known as beta testing. It is generally accepted that both alpha and beta testing are conducted on behalf of the developer, rather than the end user (Maciaszek & Liong, 2005, 20-21). However as beta testing is usually conducted by end users it is possible for the developer to gather feedback regarding the software and any changes that may be required. While the primary aim of deployment is to bring the component parts of the system together, it is likely that this will also occur alongside end user training and the creation of user documentation. (Maciaszek & Liong 2005, 20-21)

Operation and Maintenance are generally considered to be the final phase of most SDLC. Operation can be considered as the point where the software enters use as part of the day to day operations of the client. If the new software is a replacement for a previous system, it is possible

that both systems will be run in parallel for a limited time. This provides a viable alternative should the new software fail (Maciaszek & Liong 2005, 21).

Maintenance will typically begin at the same time that the software becomes operational. In software development, maintenance is normally planned and budgeted for at the beginning of the SDLC. It is worth noting that, in terms of software development, maintenance is not limited to repairing unexpected problems. Instead, it can be considered as three branches; corrective, adaptive, and perfective. Corrective fixes errors discovered during operation, adaptive modifies the software in response to operational needs, and perfective adds new features or improves the overall quality of the software. While some traditional SDLC frameworks have a clear division between development and maintenance, it can be difficult to distinguish this with some iterative frameworks. (Maciaszek & Liong 2005, 21)

### **3.2 Software Development Lifecycle Methodologies**

In the previous section, the core phases of standard software development lifecycles were defined. The following section will define and discuss several SDLC frameworks and the advantages and disadvantages that they have.

#### **3.2.1 Spiral Model**

The Spiral SDLC model (Boehm 1986, 16) offers a combination between the linear methods of the waterfall framework and the prototyping methods of incremental frameworks (Choudhury 2011, Date of retrieval 24.04.2013). There has been argument that the Spiral Framework is actually a meta-model within which almost all other SDLC frameworks could be contained (Maciaszek & Liong 2005, 25). However, the Spiral Model is independently viable for use during the development of new software and does not require the inclusion of other SDLC frameworks or methodologies. As such, it will be considered on its own merits for use in the practical work of the thesis.

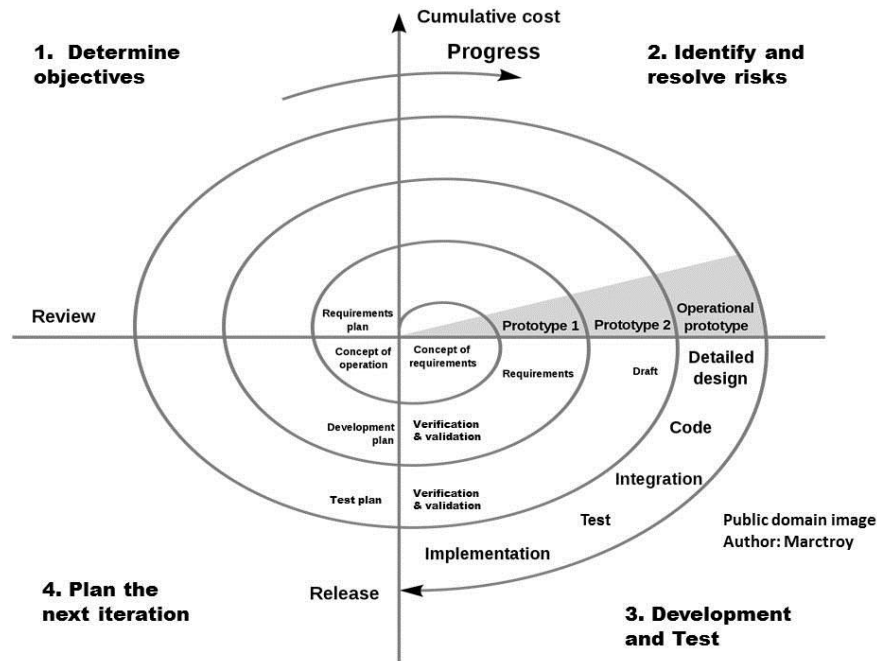


FIGURE 2. Spiral Model (Beohm 1988, 16)

As figure 2 illustrates, the Spiral Model is split into four distinct phases; Objectives, Risk Analysis, Development, and Planning. The objectives phase of the spiral model will have a variety of aims that are dependent on the development state of the software. At the beginning of the project, the objectives phase is normally concerned with the initial requirements and project planning. A risk analysis can then be performed based on these requirements and, if the project is not considered a risk, the development phase can develop the first prototype. This prototype can then be shown to the customer during the planning phase, and their feedback will then be used as a basis for the next objectives, risk analysis, and development phases (Maciaszek & Liong 2005, 25-27). Although the figure shows only two prototype stages before an operational prototype, it should be pointed out that the number of prototype releases will depend heavily on the size and scope of the software project.

For this particular SDLC, the Risk Analysis phase is greatly influential regarding the future development of the software. As Maciaszek & Liong (2005, 27) point out, risk assessment will often include the 'go, no-go' decisions for proceeding further with the project, allowing projects with developing high risk factors to be terminated without regard for investment to date.

The spiral model offers a number of advantages for software development. Risks are repeatedly analysed and revisited during the creation of the software. This can act as a safeguard against

the project failing due to unforeseen changes. Further to this, the spiral model offers a high degree of adaptability. The software is developed in small prototypes, customers are shown the system early and their feedback can be incorporated into the next requirements phase. Early exposure to the software also allows user expertise to develop alongside the software enabling easier deployment. This method of creating prototypes can also make cost estimation easier, allowing for improved budgeting (Amigo 2010, Date of retrieval 28.04.2013).

However, there are a number of disadvantages. The spiral model is best suited for large, complex projects where high cost and complex systems are involved. This is mainly due to the fact that the spiral model requires repeated investment in risk assessment that may not be viable in small or medium sized projects. The risk assessment involved in the spiral model requires that extensive skill in evaluating associated risks and uncertainties. Small and medium projects may not have personnel with these skills available (Amigo 2010, Date of retrieval 28.04.2013).

### 3.2.2 Agile

Agile is a commonly used term for a number of iterative and incremental software development methodologies. There are a number of different ways of employing agile methodologies, such as Extreme Programming or Feature Driven Development (CPrime, 2013). A recent survey by Version One (Version One 2013, Date of retrieval 20.03.2013) showed that 72% of companies using agile methods are employing Scrum or a Scrum variant. As such, this method has been selected for possible use in the practical work of this thesis.

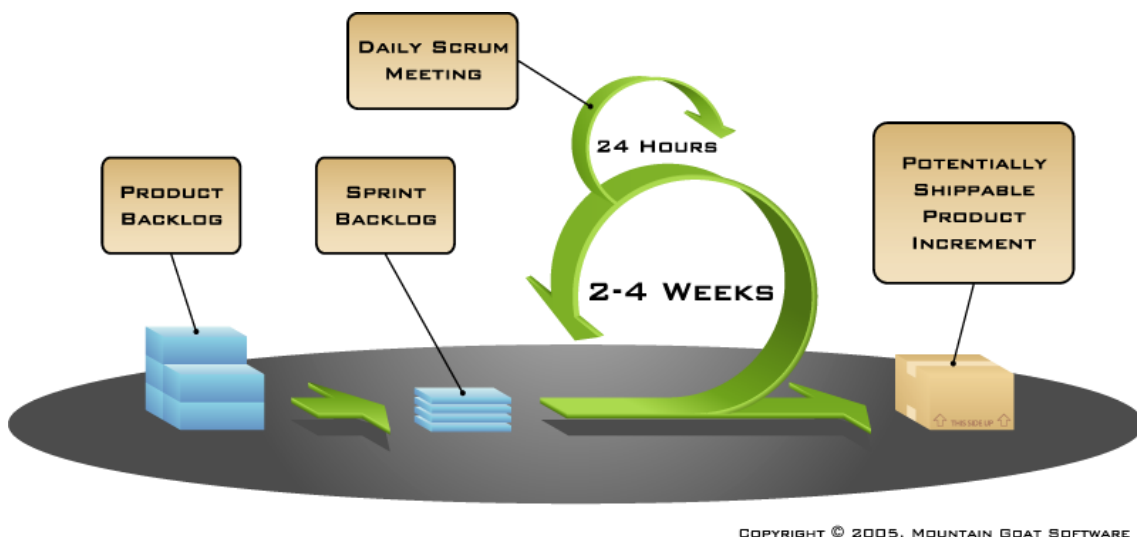


FIGURE 3. Scrum Framework (Mountain Goat Software, 2005, Date of retrieval 14.04.2013)

Figure 3 illustrates the main phases of the Scrum process and the key artefacts, or deliverables, which are produced during the development cycle. The product backlog is an ordered list of tasks to be undertaken during the project and is the source for all requirements during development. The product backlog evolves with the project and is adapted to match any changes that are required. From this, it is possible to create a smaller sprint backlog. A sprint backlog contains all of the work that should be completed during a single sprint, normally a period of two to four weeks. During a sprint there are normally daily scrum meetings. These are points where the development team can meet and discuss the work that they have done since the last meeting, the problems they have encountered, and the work they have planned. Sprint meetings are designed to be short, often no more than ten to fifteen minutes, and are used to keep the entire team up to date with the project. The increment represents the final outcome of a single sprint and is the sum of all of the work developed during that sprint. The scrum process continues until the product backlog is complete, the project budget is finished, or the customer wishes to release the product. (Schwaber & Sutherland, 2011, 5-14)

When correctly implemented, Scrum offers a number of advantages. The methodology allows projects where requirements are evolving or difficult to quantify. Cutting edge projects can be undertaken, coded, and tested without the need to develop complex requirement documentation. This means that companies have the potential to reduce the resources required for development while decreasing development time. Also, by constantly updating the progress of the work, both through the daily scrum meetings and end of sprint meetings, a clear picture of the progress can be developed. Daily meetings enable identification of issues arising within the project, allowing the solutions to be developed quickly. Constant customer feedback helps to ensure that the software in development will match customer needs. With short sprints, it is easier to cope with requirement changes and adapt accordingly. (myprojectmanagementexpert.com, 2009, Date of retrieval 03.05.2013)

However, there are some disadvantages when using the Scrum process. Some opponents suggest that Scrum development without definite end dates suffer from a high amount of feature creep. Since the process can evolve in response to customer requests, there is a potential for project management stakeholders to continually demand extra functionality. There are also potential problems when estimating project time and resource requirements. This can be especially true when dealing with projects that have poorly defined tasks or requirements. Scrum



also tends to work best for small teams, as the time requirements for the daily scrum meetings increase with the number of team members. Project testing and quality can also be negatively impacted by Scrum. With the project developing rapidly, it can be difficult to implement a testing schedule that remains up to date. Finally, the loss of a team member can have a negative impact on Scrum development. Since there is a low volume of requirement documentation, it can be difficult for a new developer to understand where the project currently is and what it still requires (Waters, 2007, Date of retrieval 15.04.2013).

## 4 SYSTEM DEVELOPMENT

One of the initial challenges faced in selecting a suitable SDLC model for this project was that the majority of models are designed around the assumption that the software will be developed by a team of developers. As mentioned, the practical side of this thesis will be developed by a single person development team and, as such, every model contained a number of unnecessary steps.

In some regards, the spiral model appeared to be suitable for this project. The creation of iterative prototypes for customer feedback would be potentially useful. However, the focus on risk assessment was considered unsuitable for the work as it was unlikely that the risk factors faced by this project would change to any great degree over the course of development and re-analysis would represent an unnecessary investment of time. Modification of the spiral model was considered, as removal of the risk assessment phase would increase the models suitability. However, after consideration, it was decided that removing the risk assessment phase of the spiral model would represent a fundamental shift and produce a model divorced from the theoretical findings. The model also required a degree of customer evaluation and feedback that was not possible due to time constraints and other commitments. As such, the spiral model was rejected for use in this thesis.

An Agile Scrum framework appeared to be more suitable for this project. However, there were a number of modifications that were required. The team roles involved in a Scrum team, Scrum Master, Product Owner, and Team Member, was combined into a single role which encompassed all of the development duties. Further to this, the daily sprint meetings were removed from the process as they served no purpose for a single person development team. Finally, the sprint length was increased in response to a number of factors. For example, this project was not being developed as a full time job. As such, project schedules and deadlines required adjustment to compensate for on-going studies. Also, arranging time to meet with the various people involved from OSAO required forewarning to ensure availability. This meant that the deliverables for each sprint required enough functionality or content for customer feedback. However, even with these modifications, the Agile Scrum SDLC appears to provide a suitable methodology for a project of this nature.

## 4.1 Preliminary Work

Prior to the first formal meeting with the thesis commissioner's team, there were informal discussions with project manager Seija Rannikko regarding the project, the current process, and the system that the commissioner wanted. These discussions provided the earliest framework regarding the requirements of the project. During November, 2012, the first formal meeting with the commissioner was held. This meeting outlined the current problems faced by the DSHC staff using the current system. Although the commissioner had a basic idea of the system they would like, a lack of technical expertise meant that they were unaware of alternatives to the current processes and available IT solutions.

The initial project proposal from the commissioner was discussed and then trimmed down to make it suitable for a thesis work. As such, it was decided that the initial project would focus on a small number of key areas. These included multi-user access to information, an easily maintainable method of storing company information, and a reduction in the number of manual calculations required. The commissioner also requested that the system be expandable at a later date, to allow for more features to be integrated by future projects. Further to this, a meeting was arranged with Lecturer Tuula Anttila to observe the current process. These observations enabled the identification of specific problems, such as the manual calculation of student ages, and allowed further discussion regarding potential solutions. The current system, and issues relating to it, was discussed earlier.

A further point of discussion regarded the provision of hardware and software for the project. Software selection and languages were left to the developer's discretion. Hardware supply, specifically the provision of a server to host the new system, would be the responsibility of the I.T. Department of the Oulun Seudun Koulutuskuntayhtymä (English: Oulu Region Joint Authority for Education).

## 4.2 Product Backlog

With the project requirements gathered, it was possible to begin creating a product backlog. Primarily, this was achieved using user stories to describe functional requirements. User stories are used to describe a single user in relation to a single desired action. Generally these can be described in terms of “As a <role>, I want to <action>, so that <result>”. An example of this is shown in Figure 4.

As a ...	<b>Admin</b>
I want to	<b>Edit student details</b>
So that	<b>I can keep student information up to date</b>

FIGURE 4. An example User Story

The product backlog also included database design tasks as well as design task for the look and feel of the website. This allowed the creation of a project plan, shown in Figure 5, incorporating four sprints. The aim was to divide the work into manageable sections with deliverables. To allow for feedback, the end of each sprint would produce a deliverable that could be taken to the commissioner’s project manager for review, For example, the end of sprint 2 would see the first prototype develop. This would allow the commissioner to view the work in progress and request changes.

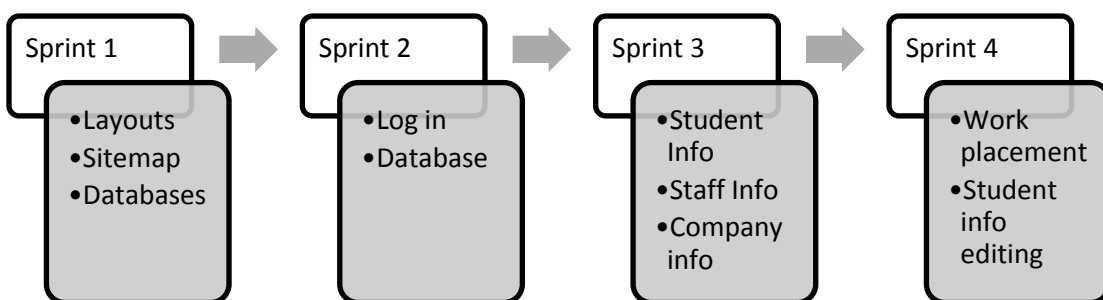
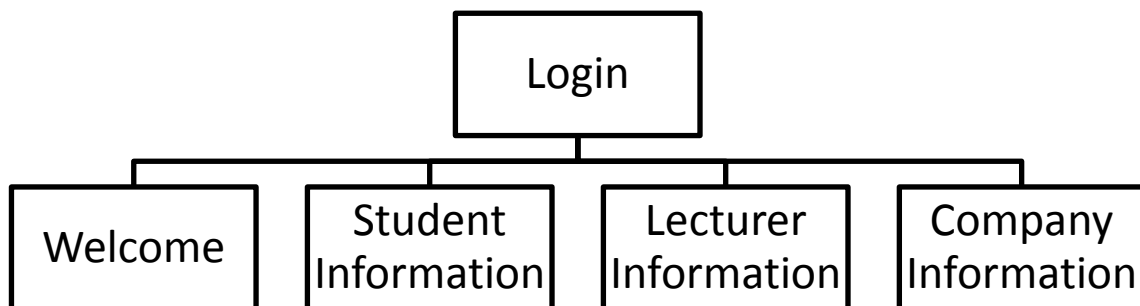


FIGURE 5. Initial Project Plan

## 4.3 Sprint 1

The first sprint focused on the design aspects of the project. Essentially, this was split into two parts. The first half dealt with the look and feel of the website and a preliminary site map. Although the commissioner indicated that functionality was of greater importance than

appearances, website usability can be influenced by good design. The majority of the design process was achieved using wireframe models, allowing for rapid creation of multiple designs. Also during the design process, it was decided that the project would utilise the Grid 960 CSS system. Grid 960 is a pre-built CSS framework designed to facilitate easy web site creation. It is lightweight, compatible with most major web browsers, and it is free (Smith, 2012, Date of retrieval 03.03.2013). The aim was to reduce the workload required for actualising the web site. The site map, Figure 5, was created to offer an overview when creating the website.




*FIGURE 6. Site map*

The second half of the sprint focused on the database design. The first class diagrams were created for the database tables which contained an outline of the data each table would contain and provided an overview of the structure and relationships of the database. The end of the first sprint involved a short meeting with Lecturer Seija Rannikko to discuss the designs and to receive feedback.

## 4.4 Sprint 2

The second sprint was dedicated to the creation of the database, the login page, and the welcome page for the website. This was significant, as it not only started the prototype creation process but was also the first point where one of the problems with the current system was solved. Specifically, this was providing multi-users access to the information system. To achieve this, the users were split into two groups, users and admins. Every user would be provided with a unique user name, password, and user type. The aim was to develop a system whereby multiple users could view access the website but only limited number could modify the information.

Creating the login page was not a difficult task. A combination of HTML and CSS was used to achieve desired look for the webpage which is shown in Figure 7. This was then combined with PHP to allow for the database connectivity and information retrieval.



The image shows a web browser window displaying a login page. At the top, there is a blue header bar. On the left side of the header is the OSAO logo, which consists of a stylized white house-like shape with three vertical bars. To the right of the logo, the text 'OSAO' is written in white. Further to the right, the text 'TYTY' is written in a larger white font, and below it, 'Työssäoppimispaikkojen työkalu' is written in a smaller white font. The main content area of the page is white. In the center, there is a login form with a title 'KIRJAUTUMINEN'. Below the title, there are two input fields: the first is labeled 'Käyttäjätunnus' and the second is labeled 'Salasana'. Below these fields is a button labeled 'Kirjaudu'.

FIGURE 7. Login Page

Testing the login page was also completed. These tests were to ensure that the website could connect to the database and that the user information was passed correctly. Once these tests were complete, the login page was then linked to a welcome page (Figure 8).



FIGURE 8. Welcome Page

The welcome page serves a number of functions. During operational use, the page will be a single point for information regarding the system status, such as upcoming planned outages or updates. Further to this, the welcome page acts as a gateway to all other pages. Although the navigation links are present in all pages, this is the first page where they appear. This is also true for the user information and logout option. Finally, it should be pointed out the Ylläpitäjä (English: Administration) is only visible to authorised users.

Sprint 2 concluded with another meeting with Lecturer Seija Rannikko. The feedback regarding the site was positive and the commissioner was happy to see the prototype. No changes were requested at this point.

#### 4.5 Sprint 3

Sprint 3 added a significant level of functionality to the website. Lecturer information was implemented in the database and the lecturer web page was coded. The inclusion of an edit function was planned for this sprint but was not implemented.

The implementation of the student and group information represented a significant advancement regarding functionality. Data for the students had been provided for the project and this allowed the creation of a suitable student table. Due to data privacy, it was not possible for student personal identity numbers to be provided. To compensate for this, every student was given a pseudo number which followed the guideline set for Finnish personal identity numbers. This meant that it was possible to implement a means of age calculation. This offered the first

reduction on the manual labour requirements of the current system. This was further extended to link to the group table of the database. By doing this, it was possible to call on the start dates of each of the work practice periods for a group and to calculate the age of each student at that point.

Company information was an unforeseen problem in terms of design and integration. Unfortunately, the information was not provided and time restrictions meant that it was not viable to create mock company data. As such, a means of tracking and creating work placements was not coded during this sprint.

The feedback meeting at the end of Sprint 3 involved Lecturer Tuula Anttila and Lecturer Seija Rannikko. During this meeting all aspects of the website were discussed. The design and layout of the website was approved, although it was suggested that the side bar replaced with a top bar navigation. Feedback was also requested regarding the inclusion of lecturer information. While it was felt that the main listing of information regarding lecturers may be unnecessary, there was a request that a popup linked to the contact details of examining and tutoring lectures be considered. However, this was not considered to be high priority and would only be implemented if time allowed. The structure and use of student information was also discussed. The commissioner was pleased to see the automated age calculations for students. However, it was felt that this information should be restricted to Administrators only. This would require a minor redesign of the webpage but would not represent a major investment of time. The website login procedure was also discussed. During these discussions it was suggested that a three-tier, rather than two-tier, login system would be used. This would allow the creation of a tutor user type. This user type would have access to the same information as a normal lecturer but with the added rights for updating student information and work placement information. Finally, the creation of company information and work placement creation was discussed. The commissioner was happy with the design plans for this and agreed to supply company information.

#### **4.6 Sprint 4**

Sprint 4 was scheduled to be the final sprint of this project. The aim was to provide a working prototype which contained all the major functionality and incorporated all of the requested changes. As requested, company information was provided for the database. However it was contained in over 100 documents, which made the transfer process challenging.



The minor design changes were implemented at the beginning of the sprint. Changes the display of student information ensured that student ages were only visible to Administrators. Further to this, the number of user types was increased to include a Tutor. This required some recoding of the webpages. Selection features were added to the student information page. These selection features allowed users to search for students by the first letter of their surname or by group (Figure 9). The alphabetical selection list was designed so the letters present would reflect the names i.e. if no students with names beginning with 'E' are present then that letter does not appear.

The screenshot shows the OSAO TYTY web application interface. At the top, there is a blue header with the OSAO logo and the text 'TYTY Työssäoppimispaikkojen työkalu'. Below the header, there is a navigation menu with links for 'Opiskelijat', 'Opettajat', 'Työssäoppimispaikat', 'Ylläpitäjä', 'Ryhmiä raportit', and 'Opettajien raportit'. Below the navigation menu, there is an alphabetical selection list with links for 'A', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'R', 'S', 'T', 'V', 'Ä', 'Ö', and 'Kaikki'. Below the selection list, there are links for 'testGroup001' and 'testGroup002'. At the bottom, there is a table with the following data:

Nimi	Ryhmä	Sähköposti	Ikä	Muodkaa
<a href="#">Antila, Aapo</a>	testGroup001	s1anaa00@students.osao.fi	21	<a href="#">Muodkaa</a>

FIGURE 9. Student information with selection options

Creation of work placement information was the most significant function added during this sprint. This function allowed the selection of a company, supervisor, and examiner for a single work placement. Figure 10 shows the summary page that shows all of the information pertaining to the work placements of a single student. The information regarding start and end dates is taken from the group information. This allows the calculation of the student's age at the beginning of the work placement.

OSAO
TYTY  
Työssäoppimispaikkojen työkalu

Olet kirjautunut pääkäyttäjänä [Lopeta](#)

[Opiskelijat](#) [Opettajat](#) [Työssäoppimispaikat](#) [Ylläpitäjä](#)  
[Ryhmiin raportit](#) [Opettajien raportit](#)

**Aapo Antila**  
**testGroup001**  
**Placement 1**

Alkaa	Loppuu	Ikä	Työssäoppimispaikat	Ohjaaja	Valvoja	
01-08-13	08-08-13	21	Mikeva / Bella Rosa	Mervi Kaikkonen	Anja Kauppi	<a href="#">EDIT</a>

FIGURE 10. Workplacement Summary

Further to this, a group summary page was implemented. This generates a full listing of all work placements for a single group (Figure 11). Similar code was used to create the lecturer summary page which covers all of the work placements that the lecturer is involved in.

OSAO
TYTY  
Työssäoppimispaikkojen työkalu

Olet kirjautunut pääkäyttäjänä [Lopeta](#)

[Opiskelijat](#) [Opettajat](#) [Työssäoppimispaikat](#) [Ylläpitäjä](#)  
[Ryhmiin raportit](#) [Opettajien raportit](#)

Valitse ryhmä :

testGroup001

Koodi	HOHU
Nimi	Aapo Antila
Alkaa	010813
Loppuu	080813
Työssäoppimispaikka	Mikeva / Bella Rosa
Osoite	Isopurjeentie 3
Postinumero	90500
Postitoimipaikka	Oulu
Kontaktihlö	Tiiminvetäjä Meeri Eerola 04478003140447800752 meeri.eerola@mikeva.fi
Kontaktihlö	Opiskelijavastaava Elina Ervasti elina.ervasti@mikeva.fi

FIGURE 11. Example Group Report

The final function added was information modification tools for the Administrators and Tutors. This allowed student, placement, and company information to be modified by authorised users and answered offered a solution to the problem of information maintenance. Finally, the text of the website was translated into Finnish.

#### **4.7 System Overview**

The system created during this thesis offers a number of solutions to the challenges faced by the commissioner. All of the highest priority functionality has been implemented and the system is ready for use. One of the major functions requested by the commissioner was a multi-tiered access for users. This was achieved with the creation of three users types; Administrator, Tutor, and Lecturer.

Administrators have the highest level of access and can edit, delete, and create all information used by the system. Tutors can view all the information but have no editing rights for the majority of the information. However, as requested by the commissioner, Tutors can assign and edit work placements for students. Lecturers have the lowest level of access to the system and may only view information.

## 5 CONCLUSION AND DISCUSSIONS

The thesis was commissioned to develop a business information system and record the development process. Overall, this goal was and the majority of functionality requested by the commissioner has been implemented. The work was challenging and interesting.

In the main, I am satisfied with the way that the project went. The final system developed matched my expectations and I was pleased to be able to develop a system which offered the commissioner viable solutions to a variety of challenges. The framework selected for developing the system proved to be well suited to the task. The changes that were required to the Agile Scrum method made it more suitable for a single person development team, while the sprint cycles allowed the work to be divided into manageable parts.

There are, however, some things that could have been done differently. Time management became an increasingly worrying factor during the thesis process. This was caused mainly by factors outwith of the thesis project and underestimations of the schoolwork I would face during the thesis process. The information acquisition from OSAO could have been handled more efficiently, as a large percentage of the information supplied was contained in multiple documents. This required a degree of processing before it could be put to use.

During the thesis process, I was able to improve my skills in both project management and system creation. Developing a system for an actual user, rather than as an academic exercise, introduced a number of new factors. Communicating with clients to produce a valid requirement analysis proved to be challenging. This was especially true in this project, as the majority of staff do not speak fluent English. This meant that any communication required a translator and added a slight complication during discussions. However this did not present a major impediment to completion of the system. Learning how to implement the requirement to the satisfaction of an end user was also a learning experience. Prior to this, the majority of projects I have been involved with have been without external influences. However, communicating and actualising the needs of the customer was a satisfying experience.

There are a number of areas where future research could be conducted. From a theoretical viewpoint, the creation of an SDLC specifically for single person development could be considered as the majority of frameworks are focused on teams of developers. This could potentially benefit students who are developing software as, in my experience, this tends to be

very unstructured. In terms of the practical work, the system has room for expansion and further automation. There are a number of functions that were removed from the original requirements to ensure that it would represent a suitable volume of work for a Bachelor's Thesis. As such, there is potential for future projects to expand upon this system. For example, direct or automated connection with the PRIMUS database could be beneficial.

I would strongly suggest that any further project work be undertaken by Finnish speaking students. I have been very fortunate to have had the help of Kirsi Rannikko-Korhonen and Timo Leskelä throughout the thesis process. They have provided invaluable help as translators for documentation and meeting. I would like to take this opportunity to express my most sincere thanks.

I have learned a number of valuable lessons during the creation of this thesis. I have improved upon the skills I have gained during my time studying in the Oulu University of Applied Sciences. It has also been an opportunity to see the practical application of academic skills. I would like to close by expressing my thanks to all of those who have helped me during this project.

## REFERENCES

- Amigo, G. (2010). *Buzzle.com/articles/Spiralmodeladvantagesanddisadvantages*. Retrieved April 28, 2013, from Buzzle.com: <http://www.buzzle.com/articles/spiral-model-advantages-and-disadvantages.html>
- Boehm, B. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, 14-24.
- Cambridge University Press. (2013, April 15). *Cambridge Dictionary Online*. Retrieved April 30, 2013, from <http://dictionary.cambridge.org/dictionary/business-english/information-system>
- Choudhury, A. (2011). *Software Development Lifecycle*. Retrieved April 24, 2013, from Software Development Lifecycle - Spiral Model: <http://www.sdlc.ws/spiral-model/>
- CPrime. (2013). *CPrime*. Retrieved April 28, 2013, from CPrime - Help - What is agile scrum definitions: <http://www.cprime.com/help/what-is-agile-scrum-definitions.html>
- Flynn, D. (1998). *Information Systems Requirements: Determination & Analysis*. Maidenhead: McGraw-Hill Publishing Comapny.
- Maciaszek, L. A., & Liong, L. B. (2005). *Practical Software Engineering*. Essex: Pearson Education Limited.
- Mountain Goat Software. (2005). *Mountaingoatsoftware.com*. Retrieved April 14, 2013, from Mountaingoatsoftware.com - scrum - figures: <http://Mountaingoatsoftware.com>
- myprojectmanagementexpert.com. (2009). *myprojectmanagementexpert.com*. Retrieved May 3, 2013, from myprojectmanagementexpert.com/advantagesanddisadvantagesofagiledevelopment: <http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-scrum-software-development.html>
- Schwaber, K., & Sutherland, J. (2011). *Scrum.org*. Retrieved May 10, 2013, from Scrum.org. - Scrum Guides: <http://www.scrum.org/Scrum-Guides>
- Smith, N. (2012). *960.gs*. Retrieved March 3, 2013, from 960.gs: 960.gs

Software Testing Fundamentals. (2012a, April). *Software Testing Fundamentals - Black Box Testing*. Retrieved March 15, 2013, from Software Testing Fundamentals: <http://softwaretestingfundamentals.com/blackboxtesting/>

Software Testing Fundamentals. (2012b, April). *Software Testing Fundamentals - White Box Testing*. Retrieved March 15, 2013, from Software Testing Fundamentals: <http://softwaretestingfundamentals.com/whiteboxtesting/>

Software Testing Fundamentals. (2012c, April). *Software Testing Fundamentals - Grey Box Testing*. Retrieved March 15, 2013, from Software Testing Fundamentals: <http://softwaretestingfundamentals.com/grey-box-testing/>

The United States Department of Justice . (2003, January). *The Department of Justice Systems Development Life Cycle Guidance Document*. Retrieved March 20, 2013, from The Department of Justice: <http://www.justice.gov/jmd/irm/lifecycle/table.htm>

Version One. (2013). *Versionone.com*. Retrieved March 20, 2013, from Versionone.com - State of agile survey results: <http://www.versionone.com/state-of-agile-survey-results/>

Waters, K. (2007). *Allaboutagile.com*. Retrieved April 15, 2013, from Allaboutagile.com - Disadvantages of agile development: <http://www.allaboutagile.com/disadvantages-of-agile-development/>

Wikipedia.org. (2013, March). *Wikipedia.org - Wiki - National Identification Number*. Retrieved May 24, 2013, from Wikipedia.org.