Project Specification and Plan

Contents

Problem Definition

"Gaming has blasted its way into the mainstream to become a multi-million dollar business, rivalling the film and record industries." - Quoted from reference [6]

The project will research and show how the expanding field of computer games design and implementation is related to Computer Science concepts and techniques. A game will be created as an example, ensuring it is clear how the game relates to this field. The development of the example game must show how Software Engineering techniques can be applied to game creation. At the end of this project an opinion on whether or not these techniques are helpful will be given.

In addition to the above there are also requirements which the developed game will meet as explained after a context survey of similar work has been performed. In general the game must be original yet fun to play and not require any special or high-quality hardware to run.

Main Objectives

The main objectives of the project are listed in order of priority.

1. Research and explain the emergence of computer game design and implementation as a branch of Computer Science. Explain how this relates to the game that will be created giving terms of reference that relate its development to Computer Science.
2. Show how Software Engineering techniques can be applied to game creation.
3. Give a conclusion stating whether or not the use of Software Engineering techniques was helpful in terms of game design.
4. The game created must be original in some way in terms of gameplay, style of graphics or concept, for example.
5. The game must not require any special or high-quality hardware to run.

Project Boundaries

This section lists what will **not** be addressed by the project.

1. Testing the developed game on many different types of computer with varying hardware and operating systems.
2. Illegal or unauthorized access to the game software.
3. External computer problems, such as viruses, preventing the game from running.

Due to time constraints it is not feasible to consider the following as part of the project. With more time they could be included.

1. Many modern game technologies for example, dynamic lighting, accurate in-game physics and animated 3D (3 dimensional) cutscenes.
2. Multi-player either co-operative or challenging.
3. Modification or map making tools allowing a user to modify the game.

Users

The people who will use the system are:

- The player

  They will operate the game and control the playable character while it is being played. They will have limited knowledge of computers so the game and controls must be easy to understand and use.

- The client

  They potentially represent the company who issued the problem and wish the game to be created. They are the potential distributor of the game on a non-commercial level or will adapt the game for marketing. They will distribute the game to the player or may be the player.

- Game testers

  They will perform the same role as the player but will try to break or discover faults within the game. They may have more technical knowledge than the player.

- Game designer

  They will be required to design, document and create the game. They will also be required to perform testing. They will be able to communicate with the game testers and, potentially, the client but not necessarily the player. Any maintenance to the project will be carried out by the game designer. Throughout the project the game designer will be able to release prototype versions of the game to game testers. Using these, and a questionnaire created by the designer, testers will give their opinions on the development of the game.

Context Survey

In this section I will evaluate other work in some popular game genres. It is sometimes difficult to split examples into these categories as they usually involve a slight mix of genres. Each example game was obtained from the reference with number given in [ ] after the name of the game. The references used are listed on page 16.

There are many different game genres. I consider only a few games and genres here, but there is a much more complete list of game genres, including descriptions of each, given in reference [5].

1. Strategy Games

Most strategy games involve a single user controlling multiple units, ranging from vehicles to infantry. The games are usually based around a high-level concept of war between factions, but can also be based around a small group of characters. The user must employ strategy to direct these characters to destroy the enemy team.

Example: Hovendall Tactics [1]

Good Points:

- Well made and simple main menu for game.
- Good interface to the game which always provides a way to get help.
- Being turn-based and using the mouse it is easy to control multiple characters at a time.
- Only information relative to the currently selected player is displayed on the interface to avoid clutter and avoid confusing the user.

Bad Points:

- No obvious user manual. Only available through game interface.
- User Manual long and complex without screenshots linking document to game.
- Have to click on (i) symbol for help which users might not recognise as a way to get help.

2. Action Games

Action is the broadest game genre. Any game that involves the user's quick re-actions can be considered an action game. The list of genres referred to above shows platform games as a separate genre, but I would include fast-paced platform games as action games. Platform games involve pressing keys at exactly the right time to overcome obstacles. They are often very simplistic, which is synonymous with action games.

Example: Sonic Zone [2]

Good Points:

- Easy to play without reading the user manual by using simple controls.
- Brightly coloured detailed graphics.
- Obvious objectives: To get to end of level and collect brightly coloured rings.

Bad Points:

- Heavily based on older Sonic the Hedgehog games.
- User manual does not contain pictures to link to game.
- No health so player forced to restart from beginning of level when hit by an enemy.

3. Horror Games

Horror games try to scare, horrify or shock the user in some way. This may be done by setting the game in a scary location, such as a haunted mansion, through clever use of sound or through visually disgusting enemies. Horror games often make use of elements seen in other genres, such as character development or item-collection in role-playing games.

Example: Cursed Undead [1]

Good Points:

- 3D graphics make the game look more attractive and realistic.
- Use of inventory to collect, store and use in-game items.
- Detailed game design document given with game. Contains pictures a screenshots to relate with game.
- Simple to use way of saving and loading a game by loading a game saved in one of 3 slots.
- Good interface showing all information related to player without cluttering screen.
- Can use mouse or keyboard to control playable character. It is good to have different control methods so the user can choose the one they prefer.

Bad Points:

- User manual is very short and does not contain pictures linking it to the game.
- Many scenes involving blood and gore.
- Can only be played above a set monitor resolution.
- Change in control method from keyboard to mouse when trying to load a game from "Load Game" menu. This may confuse the user.

4. Games aimed at younger players

This is not an individual genre but any game designed to appeal to younger players. They are usually simple games with unrealistic concepts. There are very few survival horror games designed for young people.

Example: Ness' Christmas Journey [2]

Good points:

- It has a colourful and well written user manual. It is divided into sections corresponding to different elements of the game. For example, there is one section that shows and explains all the in-game items which the player can pick up. It is also short, readable and understandable.
- Objects within the game are brightly coloured and are not overdone with minor details.
- Simple main menu to begin the game containing only 4 buttons, including a button to get help.
- Good use of foreground and background images to make the game seem more realistic.
- Simplistic un-animated cutscene at beginning of game is like a comic book which relates well to younger gamers.

Bad points:

- Information to do with player (for example, health) is hard to see because of transparency.
- Seemingly unnecessary questions at the start of the game slow it down.

5. Games featuring intelligent enemies

This is not an individual genre as intelligent enemies are used in almost every type of game. For this survey intelligent enemies means computer-controlled enemies that cannot pass through solid objects and that react to the character controlled by the player by, for example chasing them.

Example: Pyramid Panic [3]

Good points:

- Short, understandable user manual.
- Enemies change colour using coloured shading when they are near or have detected the playable character, giving a visual warning to the player.
- Different ways to play game from the beginning requiring the user to make decisions.
- Objects within the game are well-drawn and brightly coloured.
- Enemies change starting position within the game world each time it is played adding replayability and providing the user with slightly different challenges each time a new game is begun.

Bad Points:

- User manual does not contain pictures linking it to the game.
- I could not get the way of saving a game to work.

Game Definition

I believe 'survival' to be a game genre and it is interesting that reference [5] does not include 'survival', only 'survival horror', which implies 'survival' is a relatively unexplored genre. As such, it offers a suitable framework to work within. The game will be based on survival of a particular event. There are relatively few existing example games in this genre to prejudice the style of the game to be created. Given the restricted time frame the game does not need to be developed to a level that could be published. It does, however, need to be expandable into a full game or give ideas on which future games can be based.

Whether the player-controlled character does survive the event when the game is over has to be an absolute yes or no question without requiring more than 2 different endings. The game must educate the player in the basic concepts of survival (for example, the need to eat and drink) but warn them that this is not a detailed or complete simulation of survival. This should not heavily impact the entertainment value of the game. The game should support the collection of in-game items that can then be used by the character the player controls. It must, however, be suitable for players with little knowledge of computers. To this end it must be easy to use and understand without requiring a large user manual to be read or any training to be given. Characters and any computer controlled enemies within the game must move with some limited level of intelligence, for example, they must not pass through solid walls in the game unless intended and all enemies must be able to attack the player-controlled character.

The game does not need to be able to support multiple players. As mentioned in the problem definition the game must not require any special hardware, a high-quality or a fast computer to be run. Users may not have time to play the whole game from start to finish so a way to save and load a started game should be given where the items the player has collected are retained.

The game should not involve vivid scenes of violence or gore. Any scenes like this must be stylised or kept to a minimum to encourage younger players. The game must therefore not involve strong language.

Game Objectives

The game requirements are listed in order of priority.

1. Allow a user to control a character on the screen and move about. The character should not be able to move through any "solid" objects within the game.
2. Computer-controlled characters and enemies must move with some degree of intelligence and must not be able to move through "solid" objects.
3. The player must be able to collect, store and use in-game items.
4. The game must allow the player to learn basic survival concepts without being overly complicated. It must make clear that these are only guidelines to help survival and the game does **not** provide a complete or detailed simulation of survival.
5. There should be a way of saving game and returning to an already started game.
6. There should only be 2 different endings: Win and lose. This implies that the player can lose the game.
7. The games story should be based around survival of some event.
8. The game must not involve strong language.
9. Any depiction of violence or gore must be stylized.

Requirements Specification

1. Interface Requirements

This details any requirements on how the game interacts with the user.

1.1 The interface must be easy to use, understandable and readable by users who do not necessarily have experience with using computers. Users can also not be given special training to use the interface.

1.2 The interface must be simple since the user must not be required to read a large User Manual in order to use it.

1.3 The interface must display a message stating that the game does not provide a detailed or complete guide to survival but only gives the basic concepts.

1.4 The main menu of the interface should allow the user to begin a new game, load a saved game, get help for playing the game and exit the game software.

1.5 The game screen itself must clearly show any data that is associated with the player for example, their number of remaining lives. This must be easily visible while the user controls the playable character.

1.6 The interface must display computer-controlled enemies, items and other in-game objects clearly.

1.7 When the player's character health reaches 0, they die. The interface must show this.

1.8 When the playable character reaches the end of the game the interface must show that the game has been completed.

1.9 The game screen may indicate a way to get help on playing the game.

1.10 The interface must provide a simple way to quit the game and return to the main menu mentioned above.

1.11 The game is controlled by the user pressing keys on a keyboard and clicking with the mouse.

2. Performance Requirements

This details performance requirements on the game for example, speed and response time.

2.1 The game must always be available for use while it is installed.

2.2 When the game software is run it should not take long for the game to load.

2.3 Loading a saved game should not take long.

2.4 When the user presses a key on the keyboard, clicks the mouse or activates a button on the screen it should cause a seemingly immediate response within the game.

3. Security Requirements

This lists any requirements that will be needed if problems occur in the project and other security concerns.

3.1 There may be a fall-back plan to provide a different solution to the problem or to explain features which can be left out to achieve a solution.

3.2 The game must not interfere with other software while it is running.

4. Operational Requirements

This specifies actions that are required to be performed by the game.

- 4.1 The game must allow the user to control an on-screen character (the playable character).
- 4.2 The game must not allow the playable character to disappear off the visible area of the screen while being controlled by the user.
- 4.3 The game should provide no more than 2 possible endings and clearly inform the user when an ending is reached.
- 4.4 The game must allow the playable character to collect and store in-game items. The user must be able to view the items they have collected.
- 4.5 The game must maintain information associated with the playable character's survival for example, thirst. The use of in-game items will effect that information. This will allow the user to learn the basic concepts of survival.
- 4.6 The game is required to provide some limited degree of intelligence for computer-controlled enemies and characters so, for example, they cannot walk through "solid" objects within the game and can identify and attack the playable character.
- 4.7 The game should provide a simple way of saving and loading previous games so they can be continued. The game may retain all items previously collected by the playable character when loaded.
- 4.8 The game must hide or stylize scenes of violence and gore.
- 4.9 The game must remove the playable character when their health becomes 0 or they die and reduce the number of remaining lives by one. The playable character is then repositioned in the game screen unless they have no more lives where the game will be over.
- 4.10 The game may be able to "Pause" or temporarily stop gameplay (ie. remove control from the player and stop all enemies) when instructed by the user. The game should then be able to restore gameplay when instructed by the user.

5. Data Requirements

This details memory requirements and the amount of data the game will handle.

- 5.1 The game will include only 1 playable character.
- 5.2 The game will contain several objects (for example, walls) and images.
- 5.3 The game will include several computer-controlled characters and enemies.
- 5.4 The game will include no more than 3 playable areas due to time constraints.

6. Physical Environment

This details the physical requirements of the game for example, required hardware.

6.1 The game must **not** require any special, fast or high-quality hardware in order to be played.

6.2 The game must run on 'standard' computers with a mouse, keyboard and monitor.

7. Operational Environment

This details the non-physical requirements of the game for example, required software.

7.1 The game will run on any computer meeting the physical environment and using the latest version of the Microsoft Windows operating system.

7.2 The game may also run on different operating systems.

7.3 The game may **not** require any other special software to be played.

8. Functional Requirements

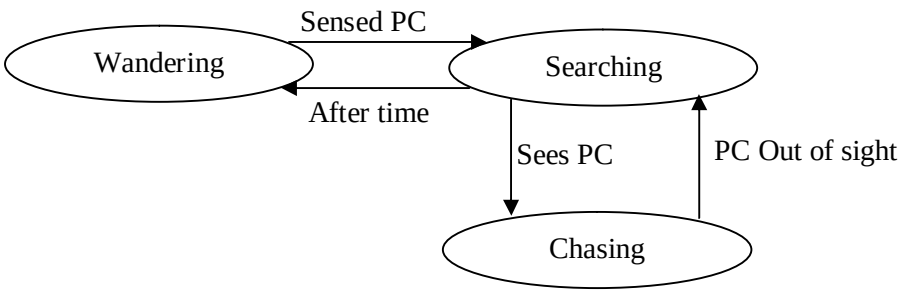
The game software must be able to do the following:

8.1 Load a menu screen allowing the user to begin a new game, load a previously saved game or exit the game. It may also provide a way to view high scores left by other users playing the game.

8.2 Allow the user to control the playable character on the screen without allowing the character to pass through solid objects in the game or disappear off the screen.

8.3 Maintain information associated with the playable character for example, thirst. This information will change while the game is being played so must be updated automatically. The information that will be maintained and displayed to the user is explained in the “Literature review and design” document.

8.4 Display computer controlled characters and enemies that have a limited degree of intelligence. These cannot move through solid objects and enemies should be able to identify and attack the playable character. The artificial intelligence of enemies should be based upon the following model.



PC = Playable character

The Wandering state is the default behaviour of the enemy. When the playable character gets close to an enemy they "sense" the character and change to the Searching state. Here the enemy moves in the general direction of the playable character but not necessarily towards them. When the enemy "sees" the playable character they switch to the Chasing state. Here they will move towards the playable character. If the playable character manages to evade the enemy and hides behind an obstacle where it can no longer be seen by the enemy the enemy will revert back to the Searching state and eventually the Wandering state if they do not see the playable character again in a fixed time. Variations of this model can be used for different computer controlled enemies.

8.5 Display in-game items that the playable character can collect. A way to store these items should be provided. The user must be able to view and use stored items.

8.6 Allow the user to quickly save a game to be continued later.

8.7 Allow the user to access the menu at any point while they are in control of the playable character.

8.8 Show a screen to indicate the user has completed the game and remove control of the playable character when they reach the end of the game.

8.9 When the playable character's health reaches 0 or they die they must be removed from the game and their number of lives must be reduced by 1. The playable character will be repositioned on the game screen unless they have no more lives where the screen indicating that the user has lost the game must be displayed and control removed from the playable character.

9. Non-Functional Requirements and Constraints

This lists other constraints on the project and software reliability.

9.1 The project is required to show and document the emergence of computer game design and implementation as a branch of Computer Science. This must be related to the game that will be created.

9.2 There is a time constraint on the project. It must be released, fully tested and documented by Tuesday the 17th of April 2007.

9.3 Other documents and logs need to be submitted to the client at various stages throughout the project and are listed in the "Deliverables and other key deadlines" section on page 14.

9.4 A meeting with the project supervisor must be held and logged each academic week of the project. These logs must be submitted to the client electronically by the end of that week.

9.5 The software of the project should be completed approximately a week before the deadline to allow it to be documented and tested.

9.6 The game should always be available for use while it is installed.

9.7 The game should run with a minimum amount of errors.

9.8 The game must be easy to use and understand by people with limited technical knowledge.

Implementation Plan

The game will be created using Game Maker (GM) which is explained in more detail at reference [4]. I chose GM because it is easy-to-use and I have used it in the past. It provides many features that are very useful for meeting the requirements listed above. For example, the embedded scripting language, GML (Game Maker Language), provides pre-written, tested and documented functions to implement artificial intelligence in games. It provides a GUI (Graphical User Interface), with drag-and-drop capabilities. It is object orientated. It works by registering objects with events and giving them actions to perform when these events occur. An example is when two objects collide and move off in separate directions. GM also provides a very easy way to define and create a game world in which objects can move around by allowing backgrounds to be placed within a game area or "room". While a game created using GM is being played the user can press 'F1' on the keyboard to automatically open a help file written by the developer. Another major advantage of Game Maker is that it has a lot of community support where I can find tutorials and ask for help if needed.

The main and only disadvantage is that GM does not provide an easy way to produce software listings or UML (Unified Modelling Language) diagrams. This is mainly because it is not reliant on a single long program file containing code. Instead the code is split between objects each containing only the functions they require.

Deliverables and other key deadlines

The project must be heavily documented. Some of these documents will only be used internally within the project (a milestone) but others will be required to be given to the client (a deliverable). This section lists those documents and any other deadlines set by the client.

All presentations are 10 minutes long, allowing 5 minutes for questions. Slides for these presentations are to be created using Microsoft PowerPoint. Each demonstration is 30 minutes long. A more detailed list of deliverables can be found in reference [7].

Type	**For**	**Date**
Deliverable	Description and Objectives document (Desc.)	8/10/06
Deliverable	Specification and Plan document (Spec.)	29/10/06
Deliverable	Literature review and system design document (Design)	13/11/06
Milestone	Detailed design document for game (Detailed design)	20/11/06
Deadline	Prototype software demonstration.	15/12/06
Deliverable	Interim Presentation and PowerPoint slides (Interim)	15/12/06
Deliverable	Revised Specification, Design and Plan documents (Revise docs.)	09/02/07
Deliverable	Complete first draft of project report (Report draft)	16/03/07
Milestone	Completed and tested software	30/03/07
Deliverable	Final project report, software and documentation including software listings (Final)	17/04/07
Deadline	Project Demonstration (Demo)	20/04/07
Deliverable	Final Presentation and PowerPoint slides (Talk)	20/04/07

I have not included the logs of supervisor meetings to be delivered to the client each week to avoid clutter.

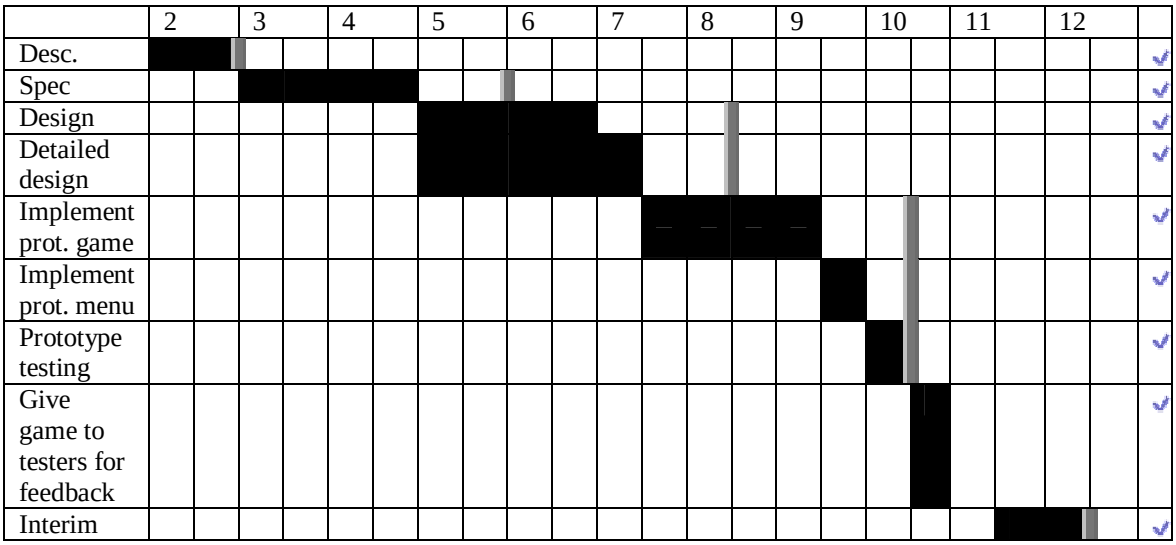
Project Monitoring Sheet

The main tasks of the project, when they are scheduled to be completed and when they must be submitted are shown in the tables below. There are weekly meetings with my supervisor to discuss the project. I have not included these on the tables to avoid clutter. I have not scheduled much time to write the project report which is a large document. I will work on this for a part of most weeks where possible.

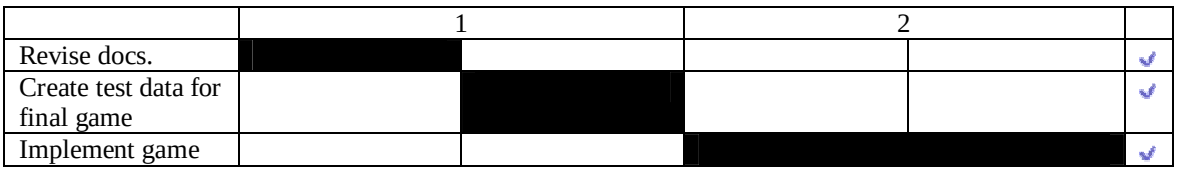
I will use a sort of depth-first approach to the way the game is created. For example, I will implement and test the game first to ensure basic gameplay and then create the main menu to allow access to the game around that.

Black bars indicate the time (in person-weeks) to spend on each task. A vertical grey bar indicates a deadline, deliverable or milestone. The final column in each table is used to show if the tasks are completed.

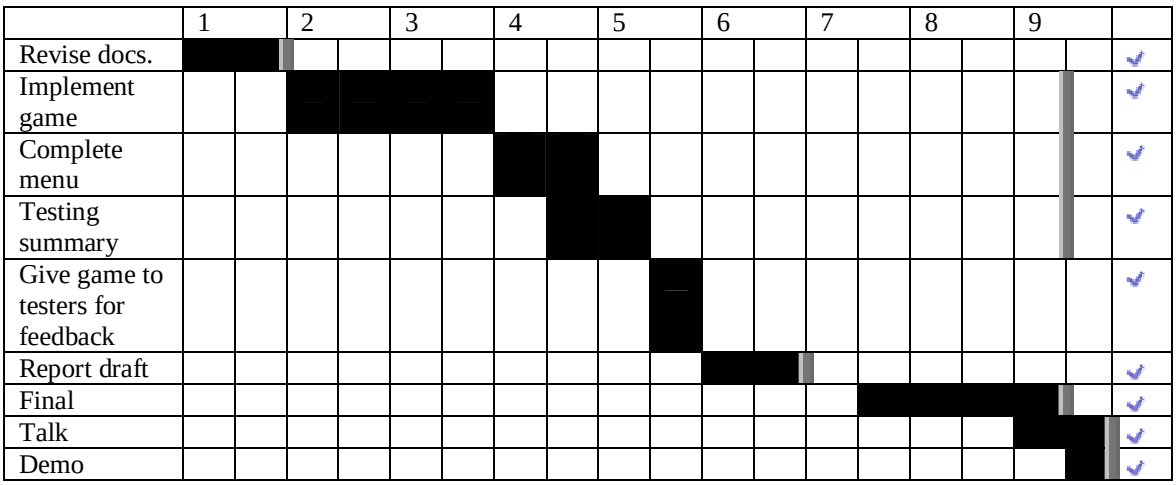
Semester 1



Weeks after exams and before semester 2



Semester 2



I have not scheduled for the spring vacation between the 24th of March 2007 and 8th of April 2007 as this is to be used as "catch-up" time for anything which is not complete.

References

[1] Title: Universiteit Utrecht - Game Design
Website URL: http://www.cs.uu.nl/docs/vakken/gds/games.html

[2] Title: Game Maker by Mark Overmars - Games Showcase: Standalone games
Website URL: http://www.gamemaker.nl/games_exe.html

[3] Title: The Game Maker's Apprentice - Game development for beginners: Companion CD
Authors: Jacob Habgood and Mark Overmars
Publisher: Apress
Publication Year: 2006

[4] Title: Game Maker by Mark Overmars
Website URL: http://www.gamemaker.nl

[5] Title: Wikipedia - Video game genres
Website URL: http://en.wikipedia.org/wiki/Computer_and_video_game_genres

[6] Title: BBC NEWS | Technology - Gaming comes of age
Website URL: http://news.bbc.co.uk/2/hi/technology/2583697.stm

[7] Title: CS4099 (major) / CS4098 (minor) Software Project
Author: James McKinna, University of St. Andrews.
Publication Year: 2006

Background Bibliography

1. Title: Software Engineering Lecture notes - Software Engineering Management
Author: Ishbel Duncan, University of St. Andrews.
Publication Year: 2005

2. Title: Xchange - A 2005-2006 software team project for the University of St. Andrews by team 1.
Website URL: http://lab-cs3.dcs.st-and.ac.uk/~jh20051/Website/Xchange.html