
SCHOLAR Study Guide

Advanced Higher Computing Science Course assessment

Authored by:

Charlie Love (CompEdNet)

Andy McSwan (Knox Academy)

Heriot-Watt University

Edinburgh EH14 4AS, United Kingdom.

First published 2020 by Heriot-Watt University.

This edition published in 2020 by Heriot-Watt University SCHOLAR.

Copyright © 2020 SCHOLAR Forum.

Members of the SCHOLAR Forum may reproduce this publication in whole or in part for educational purposes within their establishment providing that no profit accrues at any stage, Any other use of the materials is governed by the general copyright statement that follows.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, without written permission from the publisher.

Heriot-Watt University accepts no responsibility or liability whatsoever with regard to the information contained in this study guide.

Distributed by the SCHOLAR Forum.

SCHOLAR Study Guide Advanced Higher Computing Science: Course assessment

Advanced Higher Computing Science Course Code: C816 77

Print Production and Fulfilment in UK by Print Trail www.printtrail.com

Acknowledgements

Thanks are due to the members of Heriot-Watt University's SCHOLAR team who planned and created these materials, and to the many colleagues who reviewed the content.

We would like to acknowledge the assistance of the education authorities, colleges, teachers and students who contributed to the SCHOLAR programme and who evaluated these materials.

Grateful acknowledgement is made for permission to use the following material in the SCHOLAR programme:

The Scottish Qualifications Authority for permission to use Past Papers assessments.

The Scottish Government for financial support.

The content of this Study Guide is aligned to the Scottish Qualifications Authority (SQA) curriculum.

All brand names, product names, logos and related devices are used for identification purposes only and are trademarks, registered trademarks or service marks of their respective holders.

Topic 1

Project

Contents

1.1	Introduction	4
1.2	Course project	4
1.2.1	The project	5
1.2.2	Waterfall v agile development	6
1.2.3	Software design and development major	7
1.2.4	Database design and development major	8
1.2.5	Web design and development major	8
1.2.6	Introduction to your Project Documentation	10
1.2.7	Project proposal checklist	12
1.2.8	Project documentation	14
1.3	Analysis	15
1.3.1	Introduction	15
1.3.2	Description of the problem	15
1.3.3	Use case diagram	19
1.3.4	Requirements specification	19
1.3.5	Project plan	22
1.3.6	Project documentation	27
1.3.7	Summary	27
1.4	Design	28
1.4.1	Prior knowledge and revision	28
1.4.2	Introduction	29
1.4.3	Project design	29
1.4.4	User interface design	31
1.4.5	Project documentation	33
1.4.6	Summary	34
1.5	Implementation	34
1.5.1	Prior knowledge and revision	34
1.5.2	Introduction	36
1.5.3	Creating a test plan	36
1.5.4	Implementing your design	39
1.5.5	Evidence of ongoing testing	39
1.5.6	Research and development	40
1.5.7	Ongoing testing	41

1.5.8	Project documentation	42
1.5.9	Summary	43
1.6	Testing	43
1.6.1	Prior knowledge and revision	43
1.6.2	Introduction	45
1.6.3	Carrying out your test plan	45
1.6.4	Evidence of Testing	46
1.6.5	Rectifying errors and bugs	47
1.6.6	Project documentation	48
1.6.7	Summary	50
1.7	Evaluation report	51
1.7.1	Introduction	51
1.7.2	Fitness for purpose	52
1.7.3	Maintenance and robustness	52
1.7.4	Project documentation	53
1.7.5	Summary	54
1.8	Project checklist	55
1.9	Summary	56

Learning objective

By the end of this topic, you should:

- know how the project fits into the Advanced Higher Computing course;
- know what records and evidence you will be required to produce;
- be able to choose a suitable problem to tackle;
- have developed a software solution to the problem you have chosen to tackle.

1.1 Introduction

This section will help you to work your way through the project that you have to do as part of your course. It is quite unlike any other sections that you have studied so far in Computing Science.

Instead of having to learn and understand lots of new facts, concepts and skills, this section will help you make use of the knowledge and skills you already have, to:

- Analyse a problem.
- Design a solution.
- Implement the solution.
- Test the solution.
- Evaluate the completed solution.

1.2 Course project

You will discover how the project fits into the course, what records and evidence you will be required to produce, and then you will choose a suitable problem to tackle.

The Advanced Higher Computing Science course, like all similar courses, is made up of 4 sections:

- Software design and development.
- Computer systems.
- Database design and development.
- Web design and development.

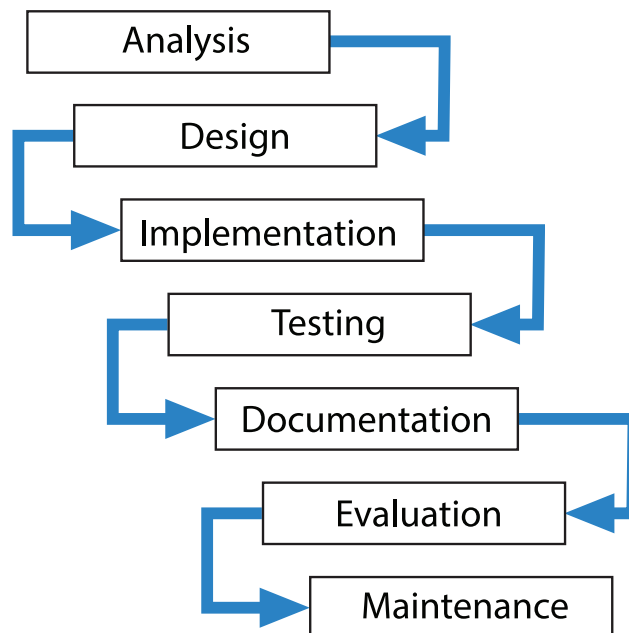
To complete the course, you need to pass the course assessment which is split into 2 parts:

1. A final exam covering all 4 sections, worth 80 marks.
2. A project, worth 80 marks.

As you work through this section, you will:

- learn about managing a project;
- choose a problem for your own project;
- complete the project;
- generate evidence for the practical outcome for this section;
- produce a project report for the course assessment.

There are several stages to work through in any software development project. You know them already:



1.2.1 The project

Breakdown of the project

There is no set time limit for your project however the guide below, of how marks will be allocated, should help you plan how much time you should spend on each section.

Analysis of the problem	10 marks
Design of the solution	20 marks
Implementation	30 marks
Testing the solution	15 marks
Evaluation of the solution	5 marks
Total	80 marks

Selecting a suitable project

The first thing you need to do is to choose a suitable project.

Your project can be based on any computing problem to which you can design and implement a software solution.

It can be on any computing problem at all, but it should:

- be at an appropriate level for Advanced Higher;
- build on learning from the mandatory units;
- be achievable.

What does "at an appropriate level" mean?

It means that the problem you choose to tackle could not simply be solved by using standard coding and algorithms that you learned for Higher Computing Science. You will need to choose something that requires more complex ideas, coming from what you have learned during the Advanced Higher course, or from research that you carry out as part of your project.

Project criteria

Your project will use at least two of the sections in the course. One will be considered the **major** part and one will be the **minor** part. Depending on which section is your major part will determine what techniques need to be used from the minor.

Key point

Your tutor will advise you about whether your ideas would be appropriate or not.

1.2.2 Waterfall v agile development

From your studies you will be aware of the traditional (**waterfall**) and **agile** development methodologies, you will now need to select which one to use in developing your project.

Before selecting which method to use you should weigh up the advantages and disadvantages of both. The following table summaries some of the key differences.

	Strengths	Weaknesses
Waterfall methodology	<ul style="list-style-type: none">• Rigid planning structure.• Good for large teams.• Helps to plan and track large software projects.• Clear agreement on outcomes at start of project.	<ul style="list-style-type: none">• Very rigid approach does not deal well with mid-project changes.• Can over-complicate simple projects.• Unidentified issues at the analysis stage can be time-consuming and costly to fix.• Little involvement of client after analysis.
Agile methodology	<ul style="list-style-type: none">• Copes well with little cumulative changes as the project progresses.• Good for small-scale projects like most Apps.• Ongoing involvement of client allows changes to be agreed quickly.• Changes cause less delay or can be tackled in the next version.	<ul style="list-style-type: none">• Works best with small teams.• Needs close version control and tracking of changes.• Can be difficult to determine the full scope of the project in the early stages.• Usually no legally binding agreement at the start.

This is going to be a substantial piece of work completed over a significant period of time, your documentation is going to be very important.

- Following the waterfall development model, each section of the documentation would be completed in order. The potential for something to be missed is less.
- When using an agile development strategy, the full requirements are not completed until the final development sprint. It is important that the analysis and design steps are followed through carefully to gain as many marks as possible.

1.2.3 Software design and development major

If you choose a software development project as the major part, you first need to decide what type of programming you wish to use, **procedural** or **object oriented** programming.

The data structures required for your project will depend on which type of language you have chosen.

In *Software design and development*, you learned about using the following programming data structures:

- Procedural programming:
 - 2D arrays;
OR
 - Array of records;
- Object oriented programming:
 - Arrays of objects.

Your project will need to use at least one of these constructs / standard algorithms.

- Sort:
 - Insertion;
 - Bubble;
- Binary search algorithm.

You will then integrate your project with either a database or a web-based application.

Database design and development minor

If you select a database application as the minor part for your software development project, it must have:

- A database with at least **one** table.
- A connection to the database that can be opened or closed from the program and an SQL query can be executed.
- The results will be formatted in the program.

Web design and development minor

If you select a web application as the minor part for your software development project, it must achieve one or both of the following:

- Use HTML/PHP to create an interface/form to allow the user to enter information/data.
- Display the formatted result.

1.2.4 Database design and development major

If you choose a database development project as your major part it must meet the following criteria:

- Create a database using SQL;
- Have at least 4 related tables;
- Uses SQL queries that use at least two of the following:
 - Subqueries;
 - A logical operator (NOT, BETWEEN, ANY, EXISTS);
 - Uses three or more tables.

Software design and development minor

If you select a software application as the minor part for your database development project, then your program must do the following:

- The program will receive its input from the database file.
- Display the data from the database in a formatted way.

Web design and development minor

If you select a web application as the minor part for your database development project, it must achieve the following:

- Use HTML to display the formatted result.

1.2.5 Web design and development major

If you choose a web development project as your major part it must meet the following criteria:

- Use of external Cascading Style Sheets (CSS) to define media queries to set different layouts for the website;
- Forms that use the following elements:
 - Action;
 - Method;
 - Name;
- Session variables;

- Server side processing to process form data and assign variables.

Database design and development minor

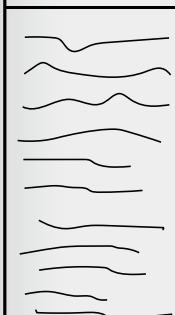
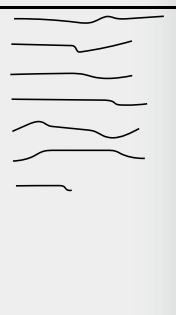
If you select a database application as the minor part for your web development project, it must have:

- A database with at least **one** table.
- A connection to the database that can be opened or closed from the program and an SQL query can be actioned.
- The results formatted in the program.

Software design and development minor

If you select a software application as the minor part for your web development project, it must achieve the following:

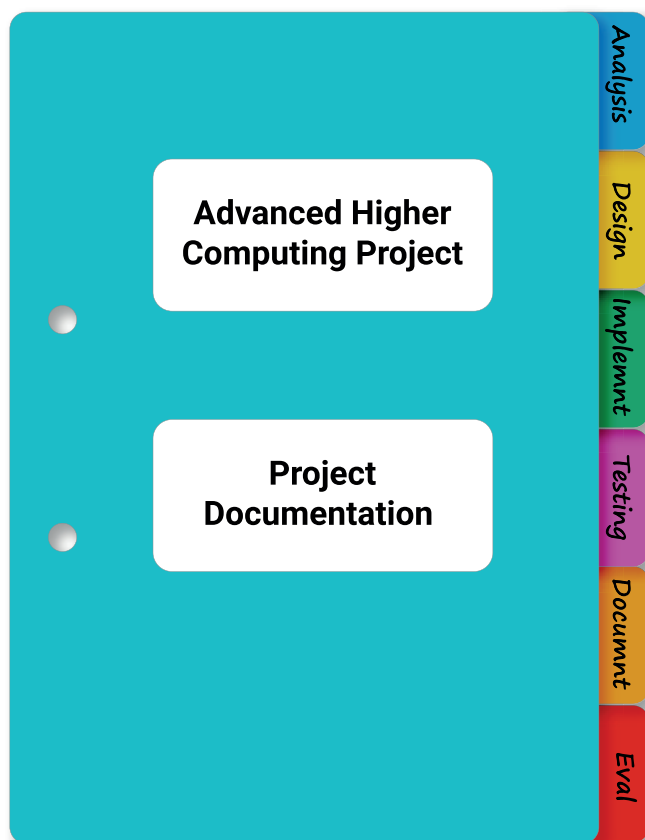
- An embedded program that makes use of the data structures looked at in the course (2D array, array of records, array of objects).

Table of Criteria	
Major	Minor
WDD	DDD
	

Name: Anne Other
Date: 01 / 06 / 2019

1.2.6 Introduction to your Project Documentation

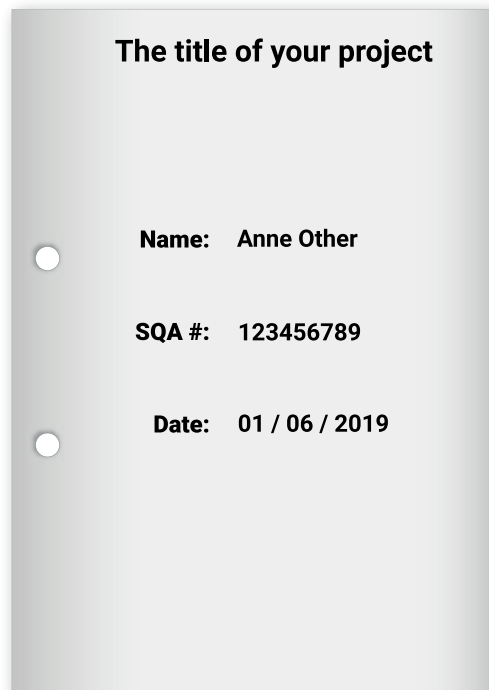
As you design, develop, test and evaluate your project, you will create documentation (typed or handwritten), notes, diagrams, screenshots, code samples, etc. All of these need to be kept, the place to keep them is in your Project Documentation. It will help keep everything together and reduce the risk of something going missing.



The example pages are simply to remind you of what to include, the content and style is entirely up to you. At the end of each topic is a contents page to act as a reminder of what it should contain.

Examples

1. Page



The title of your project

Name: Anne Other

SQA #: 123456789

Date: 01 / 06 / 2019

2. Contents

Project Documentation Contents

Project

- Project title
- Your name
- SQA candidate number
- Date

Project proposal

Analysis

Research

Design

Implementation

Testing

Evaluation report

1.2.7 Project proposal checklist

When you have settled on an idea for your project, use the following checklist to ensure it will meet all the requirements for the course.

1. My project will use two sections from the course and they are:

Major	Minor

2. What two concepts from your major part will your project use?

--

3. What role will your minor part play in this project?

--

4. Will your solution validate all inputs?

Yes / No

5. Will you be able to complete the project in the time available?

Yes / No

6. What barriers (e.g. hardware or software required, legal issues, etc) to completing this project are there?

--

7. Can you overcome these barriers?

Yes / No

8. How will you overcome these (it may be useful to have dates / milestones to keep the project on time)?

--

9. Have you discussed your project plan and the above questions with your tutor?

Yes / No

10. What comments did your tutor make?

Download the checklist: *Please go online to download the file(s)*

Add your project proposal to your Project Documentation.

Example

Project Proposal

Major	Minor

Name: Anne Other
Date: 01 / 06 / 2019

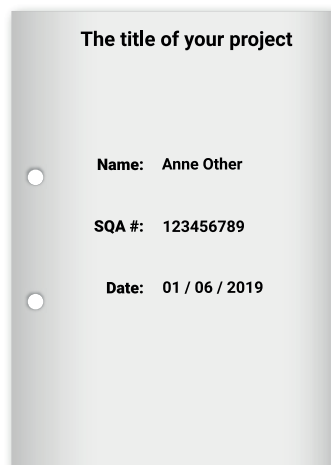
1.2.8 Project documentation

At this stage your project documentation will contain the following:

Project Documentation Contents

Project

- Project title
- Your name
- SQA candidate number
- Date



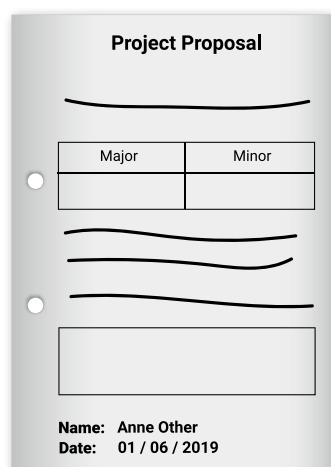
The title of your project

Name: Anne Other

SQA #: 123456789

Date: 01 / 06 / 2019

Project proposal



Project Proposal

Major	Minor

Name: Anne Other

Date: 01 / 06 / 2019

1.3 Analysis

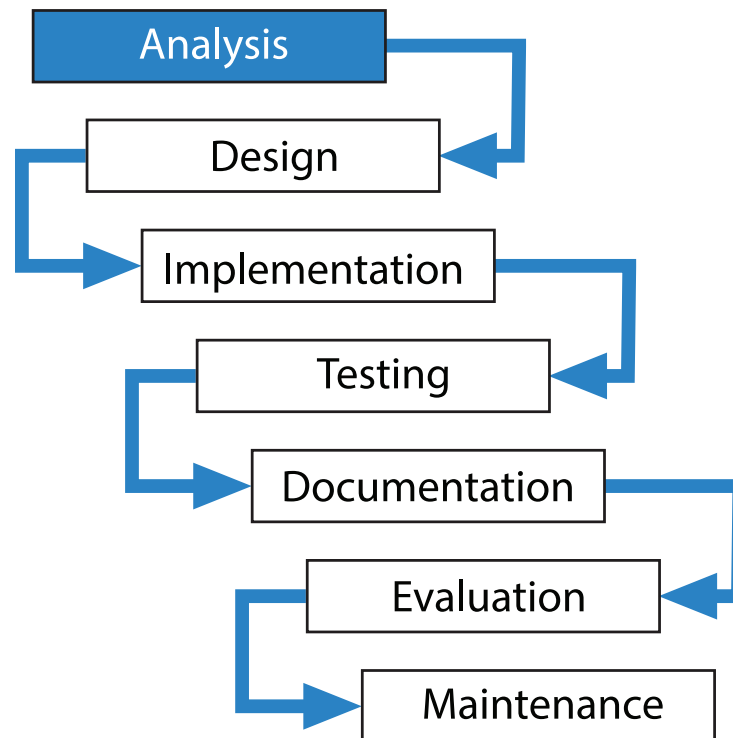
Learning objective

By the end of this topic you should have created:

- a detailed description of your project;
- a design of your project using UML;
- a detailed requirements specification;
- an overall project plan including a timeline showing your project milestones.

Evidence of these should be printed and added to your project documentation.

1.3.1 Introduction



The analysis stage of your project is worth 10 marks and is made up of the following tasks:

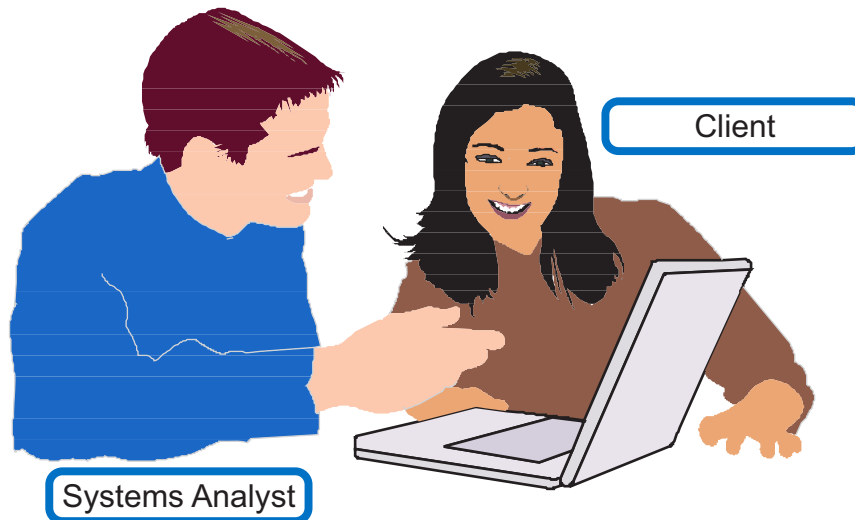
- Description of the problem (2 marks).
- UML use case diagram (2 marks).
- Requirements specification (4 marks).
- Project plan (2 marks).

1.3.2 Description of the problem

The purpose of the analysis stage is to develop a precise program or **project specification**. The starting point will usually be either an existing system which is to be improved in some way, an

outline project proposal, or a rough description of a new system required by a customer or client.

The task of refining this into a precise specification is carried out by a **systems analyst** working closely with the client. As you already know, the systems analyst (and his / her team) will use a variety of techniques to clarify the original proposal, until a precise specification can be agreed between the analyst and the client.

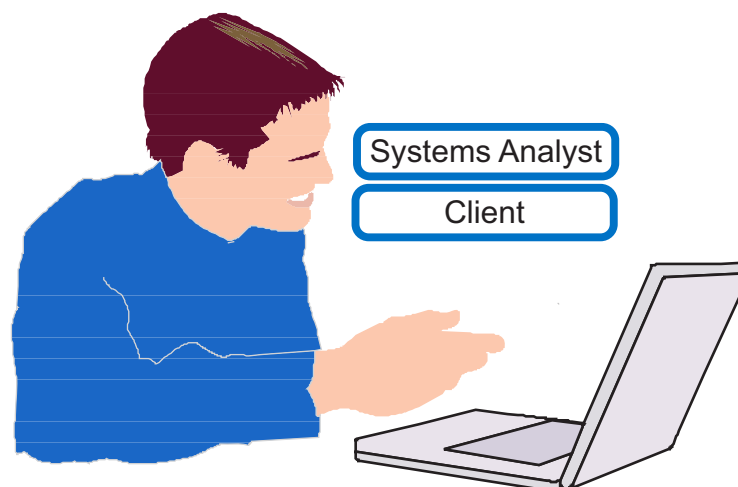


This precise specification should include:

- A clear and detailed statement of the scope and boundaries of the problem / project.
- A list of the precise functional requirements.

We will consider these in detail next.

In the context of your own computing project, you will be both the client and the analyst. You have already written down a project proposal, which gives a rough description of what your project will be about. Your next task will be to clarify that into a specification.



But first, we should take a closer look at what the specification should include.

Statement of requirements / project proposal

This is what you already have: the initial statement of requirements from the client. It is enough to get the analysis process started, but not nearly detailed enough to begin design work. In a "real-life" scenario, it will lead to many meetings between the client and the systems analysts, and may involve a **feasibility study** to ensure that the program / project is possible within the client's budget and time constraints.

For your project, this will have taken the form of a discussion between yourself and your tutor. Your tutor may have had to point out to you that your original idea was too ambitious, or that the required resources were not available, or perhaps that you could extend your idea into something more interesting or challenging.

Now it is time to turn your project proposal into a project specification.

Example

Project Proposal

Major	Minor

Name: Anne Other
Date: 01 / 06 / 2019

Scope and boundaries

It is very important at the outset to establish clearly the scope and boundaries of the project. Scope and boundaries are opposite sides of the same "coin". Between them, they give a precise description of the extent of the project.

Here is a typical statement about scope and boundaries. You will find similar statements by searching the web. "The project scope states what will and will not be included as part of the project. Scope provides a common understanding of the project for all stakeholders by defining the project's overall boundaries."

One way of thinking about it is:

- The scope clarifies what the project must cover.
- The boundaries clarify what the project will not cover.

For example, suppose your project was to develop an expert system giving students guidance on job opportunities which they should consider after graduating from University.

The scope of the project would be to create an expert system. Then it would be necessary to describe the range of jobs and degrees that would be included in the system, the level of information that would be output by the system (does it suggest contact addresses as well as simply job types), the types of questions that the user will be asked. Does it cover all degrees, or is it only for students with Computing Science degrees, and so on... All these things will define the **boundaries** of the system.

Sometimes it is also helpful to spell out exactly what will NOT be covered. So, for example, a clear statement could be made which states that the system will NOT cover advice on jobs for those with medical and veterinary degrees, or jobs overseas.

The scope and boundaries could also refer to technical issues. For example, they might state that the resultant system will run on any computer capable of running Windows 7 and above, but not on any other operating system.

Example

Scope & Boundaries

- Lorem ipsum dolor
- Sed accumsan lectus
- Ut hendrerit diam
- Nam non enim quis risus
- Vestibulum rhoncus
- Fusce sollicitudin tempus
- Sed in tortor euismod
- Mauris tristique dui et augue
- Nam a est in turpis
- Nullam lacinia elit vel

Name: Anne Other
Date: 01 / 06 / 2019

Importance of scope and boundaries[Go online](#)

Q1: Why is it important to clarify the exact scope and boundaries?

Defining scope and boundaries[Go online](#)

Now, starting from your project proposal, create a clear statement of scope and the boundaries for your project.

Top tip

Write this as a bulleted list, rather than as a paragraph. This has two benefits - it helps you to clarify your ideas, and it gives you a clear list to use at the end of your project when you are evaluating what you have produced.

Discuss this with your tutor.

Add the scope and boundaries list to your Project Documentation.

Don't forget to put your name/initials and the date on the page.

1.3.3 Use case diagram

For your project, use the *Unified Modelling Language (UML)* topic to help you create a **Use Case** diagram that shows all actors, use cases and relationships for your software solution.

UML has been used and exemplified in three sections of this course, *Software design and development*, *Database design and development*, and *Web design and development*. It can be found in **Analysis » Unified Modelling Language (UML)**.

Key point

In your Use Case diagram, indicate any connection / integration between the major part and minor part of your project.

1.3.4 Requirements specification

The requirements specification will require you to create both the **end user** and **functional** requirements for your project. It is worth 40% of the analysis marks and will be essential in helping you in the design stage.

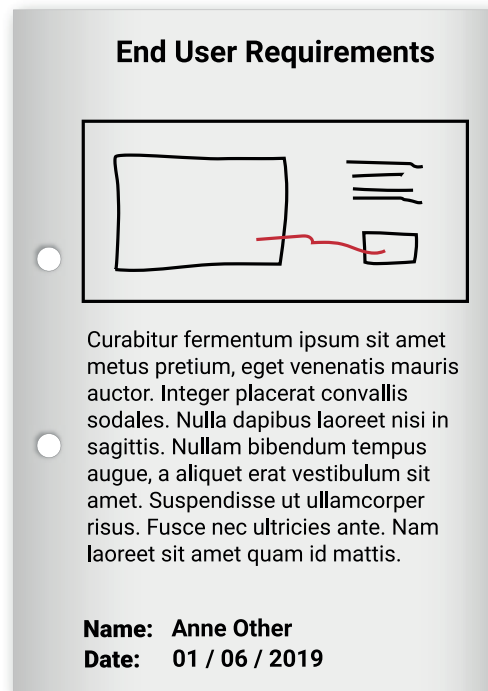
End user requirements

This will be detailed information about what the client (end user) wants or expects the software to do. The development team would make use of user surveys, interviews and scenarios to help create these. It may also be useful to observe current procedures being carried out by the client to help the developers understand the client's requirements.

You will also need to refer to the Use Case diagram to help create these.

Key point

The end user requirements should be used as the basis for developing your functional requirements.

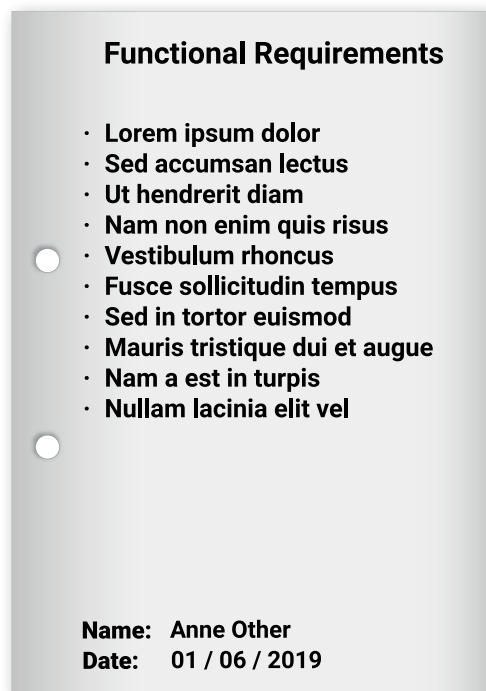
Example**Functional requirements**

The scope and boundaries define clearly the extent of the project. It is like drawing an outline map of a country. Now you have to fill in some detail inside the outline. The best way to do this is to produce a list of **functional requirements**.

Function requirements = what the product must do!

For example, if the project was to write a program to calculate the cost of sending a parcel, the list of functional requirements might include:

- Attractive welcome screen.
- All options available as clickable buttons on screen.
- User input of destination, weight and dimensions of parcel.
- User verification of all inputs.
- Output displayed on screen, and spoken through speakers.
- All colours and fonts complying with latest guidance on accessibility.
- and so on ...

Example**Key point**

Any functional requirements you create will need to be able to be tested in order for your project to be successful.

Listing functional requirements[Go online](#)

Starting from your project proposal and statement of scope and boundaries for your project, write a list of functional and end user requirements. Again, a bulleted list is probably the best way forward. As before, this has two benefits - it helps you to clarify your ideas, and it gives you a clear list to use at the end of your project when you are evaluating what you have produced.

Discuss this with your tutor.

Add the list of functional and end user requirements to the analysis section of your project documentation.

Specification review

You should now be at the stage where you have analysed the project proposal, and have a detailed and precise project specification, comprising the scope and boundaries of the project, and a list of functional requirements.

You (as systems analyst) must now agree with yourself (as client) that this is what you are going to tackle. Hopefully, you will reach an agreement! However, as you review your specification, you

might decide to add some items, or limit the scope. If so, now is the time to make these changes.

1.3.5 Project plan

Having established exactly what is required, the next stage is to develop a project plan.

The project specification states **what** is to be produced.

The project plan will state **how** you are going to do it.

There are 3 important aspects to a project plan. These are:

- Dividing the overall task into manageable sub-tasks.
- Estimating realistic times for carrying out these tasks.
- Setting up a system for monitoring progress and managing your time.

We will look at each of these separately.

Key point

Keep your project plan up-to-date as you go along and when dates change. This is an area identified by marking teams as not being fully documented which means marks can be lost in later development stages.

1.3.5.1 Sub-tasks

In "real life" software development projects, this stage not only requires identifying the different tasks to be carried out, but also allocating these task to members of the project team, or to groups within the team. You will be doing all of these tasks yourself.

As a first step, you can divide the project up into sub-tasks corresponding to the 7 stages of the software development process:

1. **Analysis** - already done
2. **Design**
3. **Implementation**
4. **Testing**
5. **Documentation**
6. **Evaluation**
7. **Maintenance** - not required for Advanced Higher

Almost certainly, you will need to add in **research** at an early stage - probably before the design stage. All the stages down to testing are required for the section assessment. Documentation and evaluation will be included in the project report for course assessment.

So the list now looks like:

1. **Analysis** - already done
2. **Research**
3. **Design**
4. **Implementation**
5. **Testing**
6. **Project Report**

The next stage is to break these down into smaller tasks. These will depend on the specific nature of your project. Here are some ideas which might be appropriate:

Research

- Investigating different ways of implementation (e.g. different programming languages or software).
- Finding out how to implement (whatever).
- Looking at similar products and considering the best approach.
- Selecting a strategy / programming language / software development environment.

Design

- Drawing sketches of the user interface.
- Creating dataflow and structure diagrams.
- Writing pseudocode for modules.

Implementation

- Creating user interface forms.
- Creating test data and a test plan.
- Coding main structure.
- Coding section 1.
- Coding section 2.
- etc ...
- Module testing and debugging.

Testing

- Component testing.
- Acceptance testing.

Project Report

- Collating evidence from your project documentation.
- Writing user documentation.
- Writing technical documentation.
- Writing an evaluation report.

Listing project sub-tasks

[Go online](#)

Create a table - with 5 columns.

Label the columns (from left to right) **sub-task**, **time**, **target date**, **completed**, **comments**.

In the left hand column, list all the sub-tasks for your project, in the order you would expect to carry them out. It should look like this:

Sub-task	Time	Target Date	Completed	Comments
Project proposal				
Specification				
Research				
...				
...				
etc				

Get your tutor to check this before you go ahead.

1.3.5.2 Time planning

Now that you have mapped out the work that is in front of you, you need to think about time-planning.

For a real-life software development project, this is absolutely vital, as the main cost for most projects is staff time. Suppose a project is being tackled by a team of 25 analysts, programmers and other staff, on an average monthly pay of £2000. The project plan estimates that the project will take 2 years to complete. The staffing costs will be £2000 x 25 staff x 24 months = £1.2 million. The project manager (from past experience!) decides to build in a 20% allowance for project slippage, so goes to the finance director asking for a staffing budget of £1.4 million.

Unfortunately, for a variety of reasons, the project takes 8 months longer than expected, even with 4 new team members added during the last 8 months to speed things up. The final staffing costs

come to £1.7 million, an overspend of £300,000. In addition, the client claims a £500,000 discount for failure to complete the project on time, so the company ends up almost £1 million worse off than expected.

Example

Project Plan

Risus	} < } < } < }
Dolor	< } < } < } < }
Ipsum	} < } < } < } < }
Lorem	} < } < } < } < }

Name: Anne Other
Date: 01 / 06 / 2019

In your case, no-one is paying you for doing your Advanced Higher project. However, if your time-planning is unrealistic, you may lose vital marks as you may run out of time to complete the work.

So, now it is time to plan your time!

Estimating project timescale

Go online



You have a table listing all the sub-tasks.

Stage 1

Go down this table, and estimate the time required for each sub-task and enter it in the second column. The times for the first 2 items will not be estimates, as you have already done them!

Add up the times. You are expected to spend around 30 hours (40 as an absolute maximum) on the design, implementation and testing, and another 10 hours or so on the project report. If your estimates fall within this range, then that is fine. If your estimates add up to more than this, you need to go back to your specification, and cut it back. Either reduce the scope of the project, or remove some functional requirements. It may be possible to identify some aspect which would be time consuming but would not contribute much to the overall mark you get, for example, if you were creating a web site, there is no point creating lots of similar pages - instead implement a couple of examples only. Discuss this with your tutor.

If necessary, go back and annotate your specification to reflect this. Then amend your time plan. (This is an example of the iterative nature of the software development process).

The final section - the project report - should be possible within about 10 hours.

Stage 2

Estimate how much time you will be able to spend each week. Remember that this will vary over the year. You might be able to spend 4 hours per week most of the time, but less than that during holidays or when you have assessments.

Use this information, and a calendar, to write a target date against each sub-task. Allow some slippage - you might be off ill for a week, or held up because some task turned out more complex than you had imagined.

Now check that the target date for the final section is in advance of when your tutor needs to receive it! If so, that's fine. If not, you are going to have to spend more time per week than you had allocated.

Once you and your tutor are satisfied that you have a realistic time plan, you can print it, and file it in your Project Documentation. Print a second copy to give to your tutor.

1.3.5.3 Monitoring and management

The time plan you have created is a vital tool for managing and monitoring your project. You should refer to it throughout the project. As you complete each section or sub-task, you can fill in the actual date you completed it. Add any comments as you go along.

After a while it will start to look like this:

Sub-task	Time	Target Date	Completed	Comment
Project proposal	2 hours	24 Aug	24 Aug	
Specification	2 hours	26 Aug	26 Aug	
Research into ...	6 hours	4 Sept	9 Sept (partly)	Most done, wrote off for more info
Selecting method	1 hour	5 Sept	10 Sept	
Design of user interface	2 hours	7 Sept		
etc				

In this example, the project is already behind the time plan by a few days, and the research has not been completed.

Does it matter? It depends!

You should have built in some extra time for unforeseen difficulties. If so, you should be able to catch up later. Keep an eye on any slippage, though, so that it doesn't become unmanageable. If you start to get too far behind, then you will need to take corrective action. That could mean spending some extra time for a couple of weeks to catch up, or discussing with your tutor whether you need to

amend your proposal. The action you take will depend on your situation and the reasons for any delay.

On the other hand, you might find you are ahead of schedule. Good! But don't become complacent. Something later on might hold you up, so you will be glad to have some time in hand.

Monitoring and managing are very important. Use the table for monitoring. Don't cheat. Be honest with yourself. And use the information to make good management decisions.

1.3.6 Project documentation

At this stage your project documentation will contain the following:

Project Documentation Contents

Project

- Project title
- Your name
- SQA candidate number
- Date

Project proposal

Analysis

- Scope and boundaries
- End user requirements
- Functional requirements
- Project plan

1.3.7 Summary

Summary

You should now have completed the following tasks:

- identified the scope, boundaries and any constraints of your project;
- created a detailed UML use case diagram that shows how your project's major and minor parts will integrate with each other;
- have a full understanding of the project requirements, both end user and functional.

1.4 Design

Learning objective

By the end of this topic you should be able to:

- describe the main elements of the design stage of the software development process
- describe aspects of good user interface design;
- create a user interface design for your project;
- explain and exemplify top-down design and stepwise refinement;
- exemplify the use of data flow diagrams, pseudocode, wireframes and structure diagrams.

1.4.1 Prior knowledge and revision

You should already know the seven stages of the software development process: Analysis - Design - Implementation - Testing - Documentation - Evaluation - Maintenance.

Design is the second stage, coming after analysis and before implementation. The starting point of the design stage is the project specification produced during the analysis stage. This specification is a clear, agreed description of the project, including scope and boundaries and all the functional requirements. From this, the project team can begin to design a solution to the problem.

Quiz: Revision

Go online



Q2: The first 3 stages of the software development process, in order, are:

- a) Design, Analysis, Implementation
 - b) Analysis, Design, Implementation
 - c) Documentation, Evaluation, Maintenance
 - d) Analysis, Implementation, Testing
-

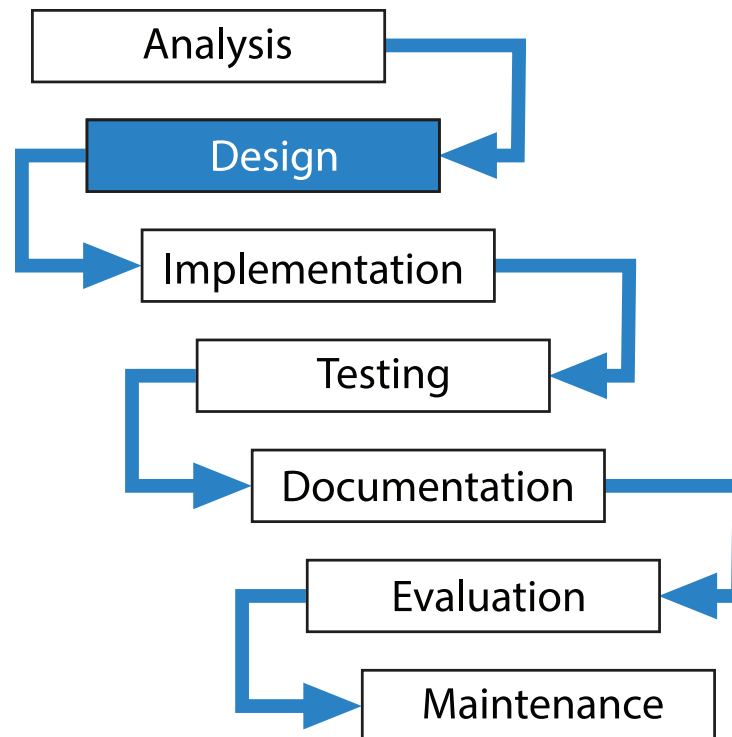
Q3: The starting point for the design stage is:

- a) the project proposal written by the client.
 - b) the project proposal agreed between client and analyst.
 - c) the detailed specification agreed between client and analyst.
 - d) the initial meeting between client and analyst.
-

Q4: The design stage is:

- a) an optional extra, which can be omitted for simple projects.
- b) essential, even in small projects.
- c) important in "real-life" project, but not required for Advanced Higher Computing.
- d) often combined with implementation.

1.4.2 Introduction



The design stage of your project is worth 20 marks and is made up of the following tasks:

- Design of Advanced Higher concepts (6 marks).
- Design of integration (4 marks).
- Design matches requirements specification (5 marks).
- Detailed user interface design (5 marks).

1.4.3 Project design

There are a number of different design notations you've studied in computing and each have their own appropriate uses for the development of a project. The type of application / system you've chosen to create will help you select the design tools that fit your project best.

Software design and development

- The standard design methodology for using procedural programming environments to develop a software application is called **top level design**. We would then use data flow and stepwise refinement to further break it down.

The use of top level design with stepwise refinement becomes even more important now that you are about to begin a major software development project.

Pseudocode, flow charts and structure diagrams could be used for any project involving coding as a major or minor part.

- If you choose to use **object oriented programming** then you would be creating a class diagram that would help you to visualise how your system will operate and will show information about the classes in your program (name, data types, methods, public and private, inheritance, constructor, array of objects).

Database design and development

- Fully labelled entity relationship diagrams (ERD) showing the entity name, type and attributes. You would show all relationship types and participation for the entities and relationships as well.
- A data dictionary is not a new concept and you will have used these at least from National 5 onwards. At Advanced Higher level, the data types used would be those used in SQL.
- Using the analysis of your project, you should be able to design all the queries that will be required for your system.

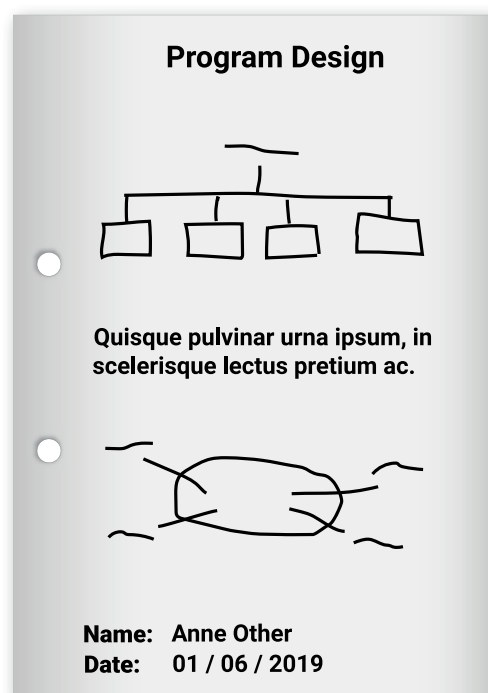
Web design and development

- Wireframes that clearly show how media queries will affect the design of your website.
- You may create some low-fidelity prototypes of the pages for your site.
- Code design which will be the same as in *Software design and development* and can be done for any JavaScript or PHP your web application will use.

Key point

There should be nothing new to learn here - you will be applying all the design and development techniques learned in the Higher and Advanced Higher courses to your project.

Example



1.4.4 User interface design

During the design stage, you will need to make decisions about the user interface and how a user would want to interact with it.

There are different styles of user interfaces you may like to consider for your project and these will differ based on the type of project you are creating.

Three examples to consider are:

- Command driven interfaces.
- Menu driven interfaces.
- Graphical user interfaces (GUI).

It is often assumed that GUIs are the best. However, that will depend on the context for your application. In some cases, a command line interface may be more appropriate. The choice of interface type will depend, not only on the user group of your application, but also on your ability and experience. Visual languages and authoring tools certainly make GUIs much easier to develop than they used to be.

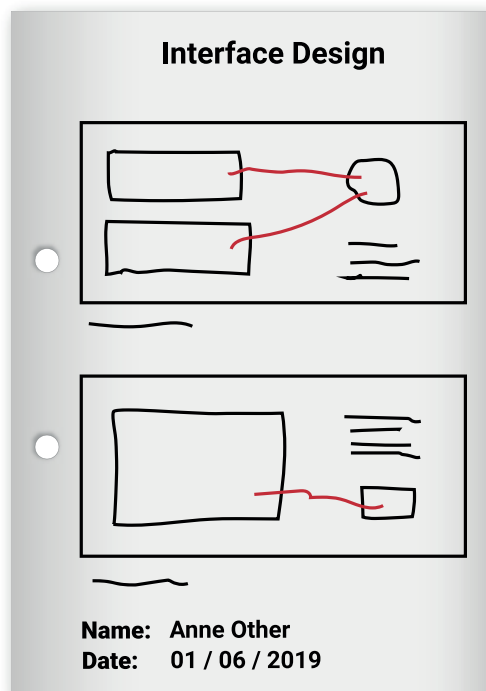
Once you have decided on the type of interface, there are many guidelines available for "good interface design". Some of these are given as follows.

Good interfaces should:

- Be as simple as possible.
- Give clear prompts to the user.
- Be consistent.
- Make sensible use of colour.
- Allow the user to "undo" the last action.
- Provide online help.

You can find many more lists in textbooks and on web sites. Different sources emphasise different aspects. You will be aware of many of these criteria from your own experience.

Add your program and interface designs to your Project Documentation.

Example**Evaluating user interfaces**

Go online



Think about some user interfaces you have used recently. They don't need to be computer-based. For example, you might think about some domestic appliances.

- Make a list of 4 different user interfaces (2 computer-based, 2 non-computer-based)
- For each one write 2 good aspects of the interface
- For each one, write 2 improvements which could be made to the interface
- Compare your answers with another student, or discuss them with your tutor

1.4.5 Project documentation

At this stage your project documentation will contain the following:

Project Documentation Contents

Project

- Project title
- Your name
- SQA candidate number
- Date

Project proposal

Analysis

- Scope and boundaries
- End user requirements
- Functional requirements
- Project plan

Design

- Structure diagrams
- UML Use Case diagrams
- Psuedocode
- Class diagrams
- Data dictionaries
- Entity relationship diagrams
- Entity-occurrence diagrams
- Query design for SQL
- Interface design using wireframes

1.4.6 Summary

Learning objective

You should now be able to:

- describe the main elements of the design stage of the software development process
- describe aspects of good user interface design;
- create a user interface design for your project;
- explain and exemplify top-down design and stepwise refinement;
- exemplify the use of data flow diagrams, pseudocode and structure diagrams.

1.5 Implementation

Learning objective

By the end of this topic, you should be able to:

- describe and create a test plan;
- complete the implementation of your design.

1.5.1 Prior knowledge and revision

You should already know the seven stages of the software development process: Analysis - Design - Implementation - Testing - Documentation - Evaluation - Maintenance:

Inexperienced developers might be tempted to plunge directly into **implementation**, but by now you should understand the importance of spending time on analysis and design before beginning implementation.

If the analysis and design has been done effectively, implementation should be straightforward. Once completed, the product can be tested and then documentation created to support users.

Sometimes, however, during implementation, problems occur which may lead to further analysis, or changes in design. Similarly, faults found at the testing stage may require certain aspects of implementation to be changed. This is why the software development process is often described as an iterative process. The seven stages of the software development process is a simplification of reality. Still, it makes sense to follow the stages in their correct order, and reduce the need to review earlier stages.

Implementation means getting involved with the computer. Choices made at the design stage are now turned into reality. This may mean using a high level programming language or some other type of development environment, like a multimedia authoring package.

Quiz: Revision[Go online](#)**Q5: Implementation**

- a) is the first stage of the software development process.
 - b) follows analysis and design in the software development process.
 - c) is the final stage of the software development process.
 - d) follows testing in the software development process.
-

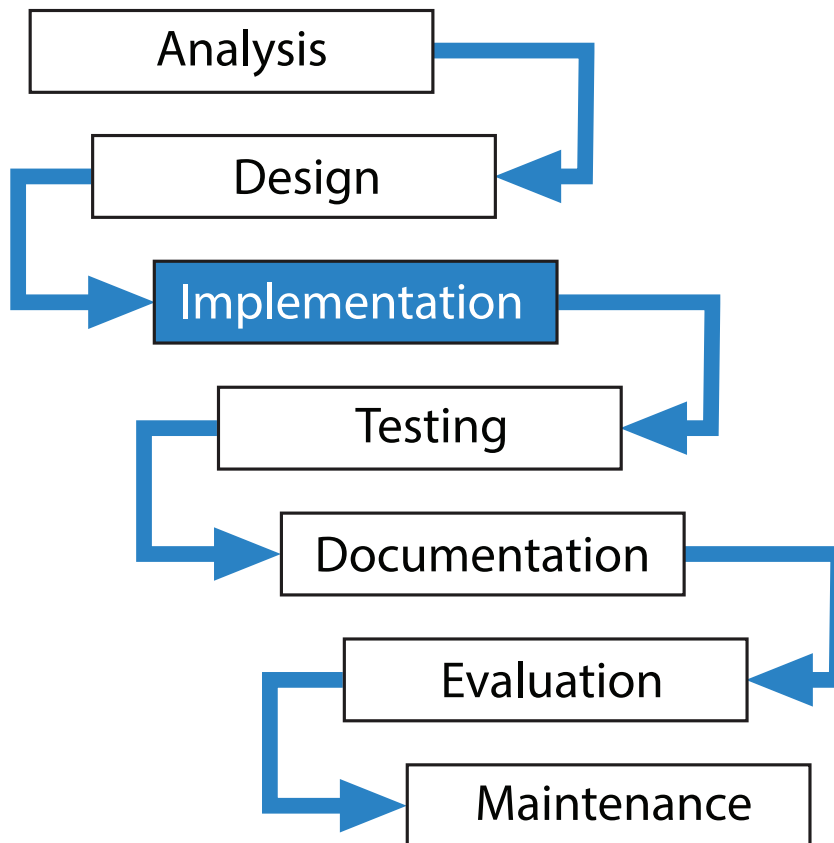
Q6: Time spent on analysis and design

- a) is time wasted
 - b) is important for AH Computing, but unimportant for "real life" projects.
 - c) can save time at the implementation stage.
 - d) slows down the whole development process.
-

Q7: During implementation

- a) you will require to make further design decisions.
- b) you will use a software development environment.
- c) you will evaluate the solution to your problem.
- d) you will certainly use a high level programming language.

1.5.2 Introduction



The implementation stage of your project is worth 30 marks and is made up of the following tasks:

- Implementation of Advanced Higher concepts in your project (12 marks).
- Integration between the major and minor parts of your project (6 marks).
- Implemented user interface from design (3 marks).
- New knowledge researched and skills developed (4 marks).
- Evidence of ongoing testing (5 marks).

Most of this topic is about YOUR project.

From the analysis and design that you have carried out, you are now ready to implement your solution. Before we get started there is one more thing to consider - the creation of a test plan.

1.5.3 Creating a test plan

You may be surprised to find that we're talking about testing now, at the start of the implementation stage. According to the 7 stages of the software development process, testing comes **after** implementation, not before it. This is correct! You can't test the program before it has been implemented.

However, a test **plan** should be created at this stage, although the plan won't be carried out until later. In "real life" major software development projects, the test plan is a lengthy and detailed document.

For your project, the **test plan** can be less ambitious!

What should be covered?

You already know that testing should be:

- Systematic (following a logical order).
- Comprehensive (covering every function defined in the functional specification).

The program should be tested with:

- Normal test data.
- Extreme test data (boundary conditions).
- Exceptional test data.

And testing can be at different levels:

- Module testing (each sub section of the program tested separately as it is developed).
- Component testing (groups of modules tested together).
- Beta (acceptance) testing (final testing of the completed program).

The exact details of the test plan will depend very much on your program, and how it is structured, and how you will implement it. However, the guidelines above will help you to structure your test plan. It should be systematic and comprehensive, and cover all three types of test data.

Key point

Keep your test plan up-to-date as you go along and especially after you have added, changed or deleted functions.

Example

Test Plan

Sub Programs

Functional Requirements

Name: Anne Other
Date: 01 / 06 / 2019

Creating a test plan for your project

Go online



List all the end user requirements.
 List all the functional requirements.
 List all the sub-tasks within your design.

Put these in tables, like these:

End user requirements	Tested	Comment

Functional requirements	Tested	Comment

Sub-tasks	Tested	Comment

Most of the sub-tasks can be tested individually as they are implemented. As you do so, tick them off, and add any comments (e.g. "works correctly" or "OK, but could be tidied up if time allows" or "doesn't work, will need to get some help").

The end user and functional requirements are more likely to be tested at the end, once you have completed all the implementation.

Some modules and functions either do the job or not. They don't depend on input data. However, many will need to be tested with a range of test data.

For any aspects of your program which will need to be tested with different types of test data, make up test data tables, giving reasons for your choice of data, and space to fill in results:

Testing ... (insert function or module being tested here)					
Type	Test Data	Reason	Expected result	Actual result	Comment

1.5.4 Implementing your design

Finally, it is now time to start implementing your design! Check back to your project plan to see what order you have planned to follow. You may decide to alter the order a little, now that you have a clear design to follow. Keep an eye on the time and target dates.

Remember to follow all the techniques of good practice you have learned in Higher and Advanced Higher **Software design and development** during the application development. These include:

- Adding internal commentary.
- Using meaningful identifiers (variables and procedure names).
- Using parameter passing and local variable rather than global variables.
- Modular programming (use existing modules if they exist).

You can save yourself a lot of work later if you keep a record of your implementation as you go along.

1.5.5 Evidence of ongoing testing

You will probably spend 20 hours or more on implementation, spread over several weeks or months. As you get more involved with coding, database queries, layout and style, etc, it can be very easy to forget about the ongoing testing you'll do, no matter how small the test may be.

You may have written a new function, unit tested it and then made changes to fix the errors, but you have forgotten the evidence / documentation to prove it.

You should create a log of your ongoing testing and update it regularly throughout the development process. Why not set aside 10 minutes at the end of every session to update your testing log.

In your Project Documentation, you should keep:

- Notes of any unit testing you carry out - these can be anything from simple comments "it works!" written on a program listing to annotated screen dumps showing part of a program working correctly with given test data.
- Note any errors you've faced, messages produced and how you fixed them. Again these can be simple screenshots and a comment explaining what has happened.

Example

Program Listing

```
// Dog class

CLASS dog IS {STRING name * 30, STRING colour *
12, REAL height, REAL length, REAL weight, STRING
breed}

CONSTRUCTOR (STRING dogName * 30, STRING
dogColour * 12, REAL dheight, REAL dlength, REAL
dweight, STRING dbreed)

DECLARE THIS.name INITIALLY dogName
DECLARE THIS.colour INITIALLY dogColour
DECLARE THIS.breed INITIALLY dbreed

FUNCTION getName() RETURNS STRING
RETURN THIS.name
END FUNCTION

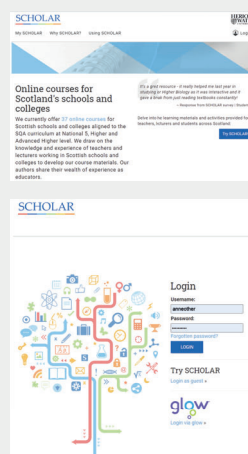
PROCEDURE setName(STRING dogName)
SET THIS.name TO dogColour
END PROCEDURE
```

* Fix

wrong variable

Name: Anne Other
Date: 01 / 06 / 2019

Screenshots



Name: Anne Other
Date: 01 / 06 / 2019

Ensure that both your project plan and test plan are filled in as you go along.

1.5.6 Research and development

Research is not meant to be a major part of your AH project, but you will almost certainly need to do some. The types of information you may need to uncover could include:

- A programming technique that you will need.
- Searching a module library for an existing module you could use.
- Finding out the needs of potential users of your system.
- Considering alternative development environments, database management software and web

development tools (e.g. Bootstrap for web design) which you could use.

- Finding out if a similar project has been undertaken.

There are many other possibilities. The methods you use will depend on the particulars of your project. Some important things to remember include:

- Be methodical about your research.
- Keep records of where you look and what you find.
- Don't waste time.

In your project you need to include a description of the research you have done, what skills you needed to develop and any new knowledge you have gained from the research.

You also need to explain how you applied the skills and knowledge you have learned to help you create the system / application.

1.5.7 Ongoing testing

While you are developing your solution, you will likely test it regularly. You need to keep a detailed log of any ongoing testing. This will help show your progress throughout the project and can be referred to during your evaluation of the project later.

This log doesn't need to be complicated, there are many tools you can use to record your testing but a simple spreadsheet would be an easy way to do it.

Your log of ongoing testing should track the following information:

- Date the testing started.
- What you are testing (Software, Database, Web).
- What issues have you found.
- What did you do to resolve these issues.
- What resources / references did you use to resolve the issue.
- Any screenshots by adding them to an appendices section.

	A	B	C	D	E	F
1	Date	What is being tested	Issues found	What have you done	Resources / references	Date issue resolved
2						
3						
4						
5						
6						

1.5.8 Project documentation

At this stage your project documentation will contain the following:

Project Documentation Contents

Project

- Project title
- Your name
- SQA candidate number
- Date

Project proposal

Analysis

- Scope and boundaries
- End user requirements
- Functional requirements
- Project plan

Design

- Structure diagrams
- UML Use Case diagrams
- Psuedocode
- Class diagrams
- Data dictionaries
- Entity relationship diagrams
- Entity-occurrence diagrams
- Query design for SQL
- Interface design using wireframes

Implementation

- Test plan
- Program listings
- Screenshots
- Research
- Log of ongoing testing

1.5.9 Summary

Summary

You should now know that:

- a test plan should be created before implementation begins;
- a test plan describes what will be tested, how it will be tested, and when it will be tested;
- good programming techniques should be applied during implementation;
- the project documentation should be maintained throughout the process;
- module testing during implementation should be recorded against the test plan.

1.6 Testing

Learning objective

By the end of this topic, you should be able to:

- carry out a test plan;
- produce a user questionnaire;
- summarise test results;
- rectify errors and bugs.

1.6.1 Prior knowledge and revision

You should already know the seven stages of the software development process: Analysis - Design - Implementation - Testing - Documentation - Evaluation - Maintenance:

You also know that the software development process is **iterative**. That means that sometimes, you have to repeat stages to get it right. This is particularly true of implementation and testing.

You know that testing should be both systematic and comprehensive. Any software should be tested under a range of conditions, and using normal, extreme and exceptional test data.

Testing should be based on a **test plan** created before implementation. The test plan should include module testing, component testing and beta (acceptance) testing.

Quiz: Revision[Go online](#)**Q8: Testing**

- a) is the final stage of the software development process.
 - b) follows implementation in the software development process.
 - c) takes place before implementation in the software development process.
 - d) is sometime called maintenance.
-

Q9: Test data designed to test the software under boundary conditions is called

- a) normal test data.
 - b) acceptance test data.
 - c) extreme test data.
 - d) exceptional test data.
-

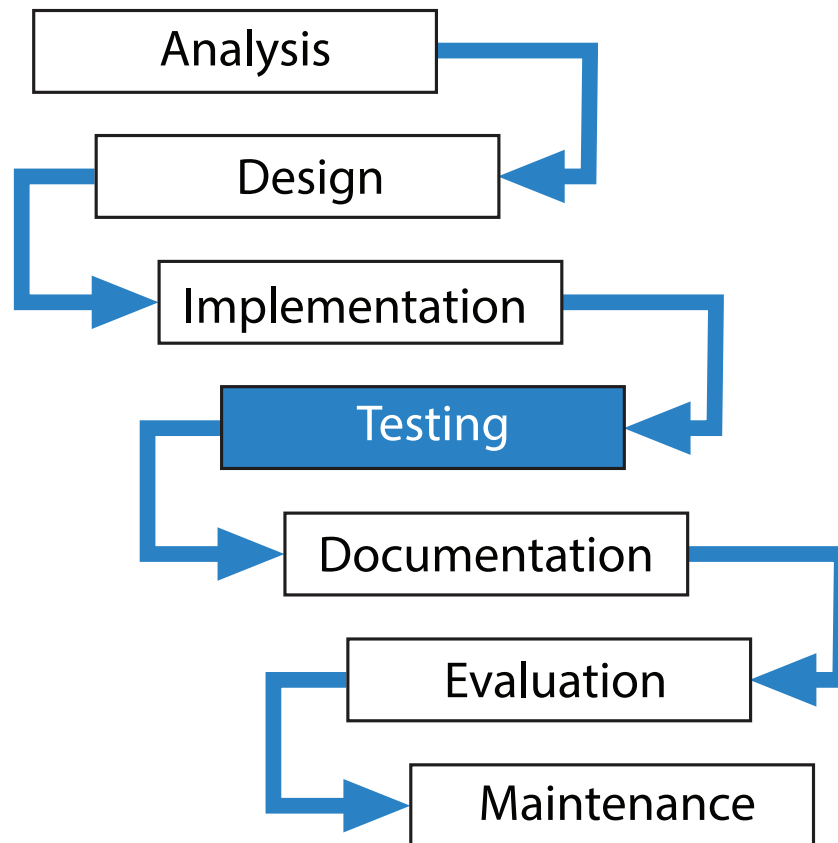
Q10: The testing of individual sub-programs as they are implemented is called

- a) component testing.
 - b) module testing.
 - c) beta testing.
 - d) comprehensive testing.
-

Q11: Comprehensive testing means

- a) testing of every possible combination of input data.
- b) testing every function defined in the functional specification.
- c) getting users to test the final product.
- d) testing with exceptional test data.

1.6.2 Introduction



The testing stage of your project is worth 15 marks and is made up of the following tasks:

- A comprehensive test plan for your finished project. (6 marks).
- Evidence of requirements testing (6 marks).
- Results of the test cases (3 marks).

Once again, most of this section is about YOUR project.

You should now have a completed project and a test plan that you devised earlier. You will apply this test plan to your project. You should already have tested each module or sub-task as you were implementing them; now you are going to test all the functional requirements in a systematic way.

1.6.3 Carrying out your test plan

Refer to your test plan in your Project Documentation. It will look something like this (except that you should already have filled in the first columns (lists of end user requirements, functional requirements and sub-tasks). Also you have probably filled in the other columns of the sub-task table, showing that you have tested each one (and corrected any bugs that you found). If there are any of these sub-tasks that you haven't tested, do so now.

End user requirements	Tested	Comment

Functional requirements	Tested	Comment

Sub-task	Tested	Comment

Now you must carry out systematic testing against all the stated end user and functional requirements which you have listed above.

Where appropriate, you should test using a range of test data (normal, extreme and exceptional).

Devising test data

Go online



If you have not already done so, create test data for each functional requirement. Usually, the best method is to create a table to display your test data and the results, like this:

Testing ... (insert function or module being tested here)

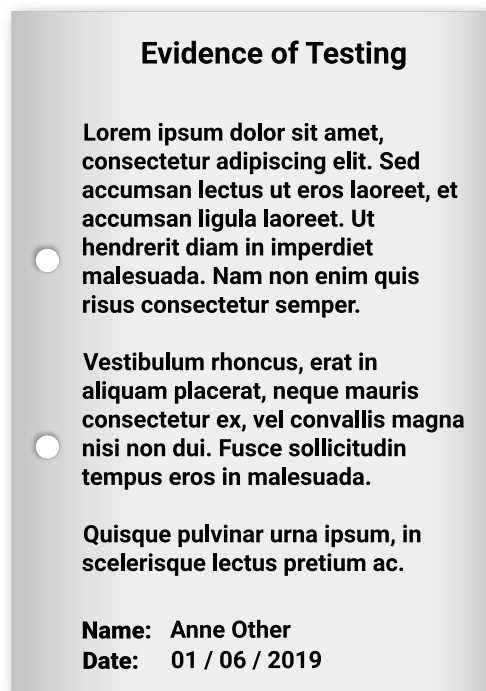
Type	Test Data	Reason	Expected result	Actual result	Comment
<i>Normal</i>	<i>5, 10, 15</i>	<i>Common input</i>	<i>"use widget B54Xa2"</i>	<i>"use widget B54Xa2"</i>	<i>OK</i>

In some cases, a table of testing is not really appropriate. The main thing is that you show what you are testing and you keep a record of the results. Remember to fill in the "expected result" column BEFORE you carry out the test(s).

1.6.4 Evidence of Testing

For your Project Documentation, you should include a description of the testing results. It is also useful to have some "real" evidence in the form of screenshots to support your descriptions. In particular, you should obtain screenshots to illustrate what appears on the screen at any stage of your development. So perhaps you might generate screenshots of the opening or title page, of an input screen, and of the output of the system. Note that there is no requirement to create screenshots of **every** test you carry out! You may test many functions of the program with many different sets of test data, but only need 3 or 4 screen shots to illustrate how the program looks on screen.

All evidence of testing - tables of results, printouts, screenshots - should be filed in your Project Documentation and carefully labelled to explain their relevance.

Example**1.6.5 Rectifying errors and bugs**

As a result of testing, you have almost certainly uncovered some errors or bugs in your project. If so, you should correct these as they are discovered. Keep a record of any corrections you make - simple notes written on the earlier print-outs may be enough. If an error requires a major rewrite of any part of the software, print it out and file it in your Project Documentation, with a hand-written note on the print-out explaining why you have made the changes. You should implement some logical use of version numbers, and make sure you label all print-outs accordingly.

Beware of the possibility of correcting a bug causing a new error in some part of the software that you have already tested. If you have applied the principles of modular programming, this should be unlikely to happen, but it is usually a good idea to repeat some earlier tests just to be on the safe side.

Sometimes, students think that they should hide and / or destroy all evidence of bugs or errors, and simply keep evidence of the completed (perfect!) program. This is not recommended! Your project will be awarded marks for your ability to correct errors, so **keep the evidence** to show that you have managed to fix any problem that occurred.

Example

Error Fixes

Amended Listing

New Test Results

Name: Anne Other
Date: 01 / 06 / 2019

1.6.6 Project documentation

At this stage your project documentation will contain the following:

**Project Documentation
Contents**

Project

- Project title
- Your name
- SQA candidate number
- Date

Project proposal**Analysis**

- Scope and boundaries
- End user requirements
- Functional requirements

- Project plan

Design

- Structure diagrams
- UML Use Case diagrams
- Psuedocode
- Class diagrams
- Data dictionaries
- Entity relationship diagrams
- Entity-occurrence diagrams
- Query design for SQL
- Interface design using wireframes

Implementation

- Test plan
- Program listings
- Screenshots
- Research
- Evidence of ongoing testing

Testing

- Completed comprehensive test plan
- Evidence of end user requirements testing
- Evidence of functional requirements testing
- Descriptions of your results

1.6.7 Summary

Summary

You should now know that:

- testing should follow implementation, and be based on the test plan;
- testing should be comprehensive and systematic;
- testing should include module testing during implementation;
- all functional requirements should be tested using a range of test data;
- acceptance (beta) testing should be carried out by independent testers;
- errors and bugs identified during testing should be rectified and/or documented;
- a summary of testing should be produced.

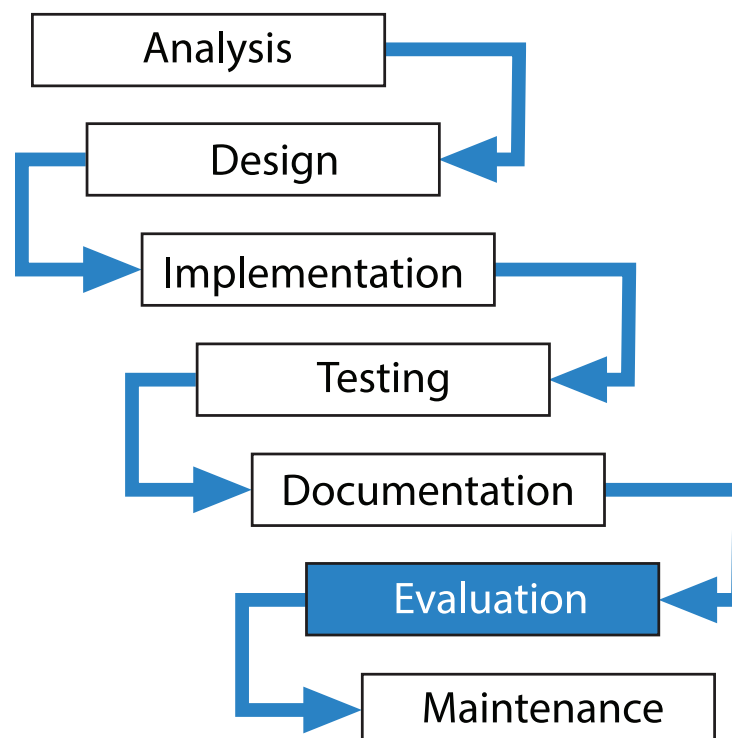
1.7 Evaluation report

Learning objective

By the end of this topic, you should have evaluated:

- if your project is fit for purpose;
- the project in terms of its maintainability;
- the project in terms of robustness.

1.7.1 Introduction



The evaluation stage of your project is worth 5 marks and is made up of the following tasks:

- Evaluating your solution in terms of fitness for purpose (3 marks).
 - How does your solution compare to the functional requirements.
 - How does your solution compare to the end user requirements.
 - The results of your testing.
- Evaluating the maintainability and robustness of your solution (2 marks).

You are now required to create a report to evaluate your completed project. You will evaluate:

1. your solution's fitness for purpose - does it do what it is supposed to do, does it work as expected;
2. how stable your system is, how easy it would be to maintain and how robust it is (e.g. can it deal with unexpected inputs).

1.7.2 Fitness for purpose

To complete this you will need to revisit the documentation from the analysis stage:

- End user requirements:
 - Using the scenarios, interviews and user surveys that were created, how would the new system work for these users?
 - Does the system work as it should for the scenarios given?
 - Were there any extra features added?
- Functional Requirements:
 - Look through your list of functional requirements, have each of these been met?
 - Explain what parts of the system meet each of these requirements.

Key point

It's equally important to be honest about the functionality of your project. If there are parts of the project that do not meet the initial requirements your report should detail that here.

1.7.3 Maintenance and robustness

You will now look at how well your system can be maintained and how it copes with unexpected errors.

You should detail all the steps you have taken to make your project maintainable and explain why it helps.

- How have you used internal commentary in your project and how has it helped?
- Did your input validation work as expected or were there changes required?
- Can you refer to your log of ongoing testing for issues that you faced and how easy were they to solve?
- What evidence do you have from the testing phase you have carried out that led to changes and how easy was it to do?
- Is your project reliable, does it do everything as would be expected every time?

Key point

Try and refer to parts of your project and show evidence - screenshots are a great way to show what changes you have made.

1.7.4 Project documentation

At this stage your project documentation will contain the following:

Project Documentation Contents

Project

- Project title
- Your name
- SQA candidate number
- Date

Project proposal

Analysis

- Scope and boundaries
- End user requirements
- Functional requirements
- Project plan

Design

- Structure diagrams
- UML Use Case diagrams
- Psuedocode
- Class diagrams
- Data dictionaries
- Entity relationship diagrams
- Entity-occurrence diagrams
- Query design for SQL
- Interface design using wireframes

Implementation

- Test plan
- Program listings
- Screenshots
- Research
- Evidence of ongoing testing

Testing

- Completed comprehensive test plan
- Evidence of end user requirements testing
- Evidence of functional requirements testing
- Descriptions of your results

Evaluation report

- Evidence of fitness for purpose
- Evidence of maintenance and robustness

1.7.5 Summary**Summary**

You should now know have evaluated your project for:

- fitness for purpose;
- maintainability and robustness.

1.8 Project checklist

Analysis	Complete
Scope and boundaries	
End user requirements	
Functional requirements	
Project plan	
Design	
Structure diagrams	
UML Use Case diagrams	
Pseudocode	
Class diagrams	
Data dictionaries	
Entity relationship diagrams	
Entity-occurrence diagrams	
Query design for SQL	
Interface design using wireframes	
Implementation	
Test plan	
Program listings	
Screenshots	
Research	
Evidence of ongoing testing	
Testing	
Completed comprehensive test plan	
Evidence of end user requirements testing	
Evidence of functional requirements testing	
Descriptions of your results	
Evaluation	
Evidence of fitness for purpose	
Evidence of maintenance and robustness	

Download the project checklist: *Please go online to download the file(s)*

1.9 Summary

Summary

You should:

- know how the project fits into the Advanced Higher Computing course;
- know what records and evidence you will be required to produce;
- have chosen a suitable problem to tackle;
- have developed a software solution to the problem you have chosen to tackle.

Glossary

Feasibility study

At a very early stage in any project development, a feasibility study is carried out to judge whether or not the project can be achieved, within any known constraints. These constraints include time, finance and availability of appropriate hardware, software and expertise.

Functional requirements

A list of all the functions which the finished program or system must be able to do. This forms a main part of the specification.

Project specification

A document agreed between the systems analyst and the client during the analysis phase, which defines clearly all the functional requirements of the systems to be developed. It forms a binding legal agreement between the developer and the client.

Systems analyst

Usually an experienced programmer, who is responsible for the analysis and design phases of software development. The Systems Analyst, working with the client, develops the project or system specification, on which the design and implementation are based.

Test plan

A plan devised before implementation of software, listing all the tests to be carried out, the test data to be used, and the expected outcomes.

Top level design

Top level design starts with an overall picture or concept of what the program should do. From this starting point, the designer can break the overall task up into smaller tasks. These smaller tasks, in turn, can be broken down into sub-tasks, and so on Eventually detailed design is applied to each sub-task. Step by step, the fine detail is worked out. The process is called stepwise refinement.

Answers to questions and activities

Topic 1: Project

Importance of scope and boundaries (page 19)

Q1:

Real world answer: Proper scope definition is critical to a project's success. It establishes the boundaries of what the project will and will not accomplish. The scope statement eliminates any confusion or ambiguity that might still exist after considering the project's goal, objectives and high-level deliverables statements. Poorly defined scope leads to "scope creep", which means that the project's objectives change as it progresses. These changes inevitably lead to increased work effort, which in turn causes project delays, cost overruns, poor team morale and/or customer dissatisfaction.

AH project answer: Proper scope definition is essential to ensure that you embark on a realistic project. If you don't define the scope and boundaries, you won't know when you have finished implementing the project, you won't be able to evaluate it properly, and finally, you will lose marks!

Quiz: Revision (page 28)

Q2: b) Analysis, Design, Implementation

Q3: c) the detailed specification agreed between client and analyst.

Q4: b) essential, even in small projects.

Quiz: Revision (page 35)

Q5: b) follows analysis and design in the software development process.

Q6: c) can save time at the implementation stage.

Q7: b) you will use a software development environment.

Quiz: Revision (page 44)

Q8: b) follows implementation in the software development process.

Q9: c) extreme test data.

Q10: b) module testing.

Q11: b) testing every function defined in the functional specification.