# Analysis

## Purpose

Feasibility Studies

* Economic
* Time
* Legal
* Technical

## User surveys

Project planning

* Scheduling
* Resources
* Gantt charts

## Requirements specification

* End user requirements
* Scope, Boundaries and Constraints
* Functional Requirements

## Purpose

The first stage of software development is making sure that you have fully understood the problem that you are being asked to solve and what your solution is supposed to achieve.

In identifying the purpose, you are defining the problem and describing what your completed solution will be able to do.

You are clearly describing what the software will be used for.

Do not forget to include a description of:

* what is it to do
* who will be using it
* why they will be using it
* the main input(s) and output(s).

Note that this is a description of *what* your solution will do rather than *how* it will do it.

At this stage, it is NOT a detailed technical description of how your solution will work, rather, think of it more like describing your solution from the point of view of the client/user with a high-level description of the overall goal and objectives.

## Feasibility Study

The creation of a new system can take months (or even years) and cost thousands (or millions!) of pounds. The basic purpose of a feasibility study is to ascertain whether such expenditure of time and money is likely to prove worthwhile and whether the objectives of the problem definition can actually be realised – what some companies want might be unrealistic and unachievable.

A feasibility study should be relatively cheap and quick to carry out. It should be complete in itself and the client does not have to commit to any further expenditure or long-term development by commissioning one – the client has no legal or contractual obligation to take developments further.

Another advantage of a feasibility study is that it can present the client with a range of possibilities from which a choice can be made. Often, several different solutions will be considered in a feasibility study before a recommendation is made.

### Economic Feasibility

This deals with the **cost implications** involved. Management will want to know how much each option will cost, what is affordable within the company’s budget and what they get for their money.

A **cost-benefit-analysis** is part of the budgetary feasibility study. If the project is not cost-effective then there is no point proceeding. Setting up a new computer system is an investment and involves capital outlay.

The costs of a new system include:

* the costs of acquiring it in the first place (consultancy fees, program development including cost of any resources required for development, etc.)
* the costs of installing it (disruption of current operations, cost of new equipment, alteration of workplace, etc.)
* the costs of maintaining it which also includes training.

In the long term, management will also want to know the ‘**break-even point**’ when the new system stops costing money and starts to make money. This is extremely difficult to quantify. However, an accurate estimate of a system’s operational life span is a valid option and will rely solely on the knowledge and experience of the systems analyst involved.



The break-even point is at the intersection of the graphs

Tangible benefits that management would certainly be looking for in the new system would be:

* reduced running costs
* increased operational speed
* increased throughput of work
* better reporting facilities

Note that not all the costs and benefits lend themselves to direct measurement. These are called *intangible benefits* e.g. new systems generally affect the morale of the staff involved, for good or ill.

### Time Feasibility (sometimes called “schedule feasibility” in older SQA docs)

Obviously, time is a main factor in the development of a new system.

Questions to be asked at this stage might include:

* \_ How long will the proposed system take to develop?
* \_ Will it be ready within the specified time-frame?
* \_ When is the best time to install?

For example, a project might have to start within six months; assuming it would take three months to purchase and install the required hardware and software and a further six months to train the end users. Such a project is not technically feasible because of shortage of time so it would not go ahead unless some of the time constraints were reviewed and changed.

### Legal Feasibility

This has to do with any conflicts that might arise between the proposed system and legal requirements. For example:

* How would the new system affect contracts and liability?
* Are health and safety issues in place?
* Would the system be legal under laws such as GDPR?
* Would the system be legal under other laws local to the UK or wider jurisdictions where your software may be used or used to store data from inhabitants of those countries e.g. EU or USA?
* What are the software licensing implications for the new system?

Software licensing can be quite a thorny problem. Licences can be purchased as:

client licence (per seat), server licence, network licence or site licence and the period of operation may be annual or perpetual. Software vendors vary in their licensing regulations so this has to be fully investigated.

### Technical Feasibility

The feasibility study must ascertain what technologies are necessary for the proposed system to work as it should. Perhaps suitably advanced technologies do not yet exist. Unless it is the object of the project to design a system to use such advanced technology, this would rule the project out as being a non-starter. It would be a foolish move for a feasibility study to evaluate technologies which are either under development or undergoing testing.

Given that suitable technology does exist, the feasibility study must establish if the organisation already has the necessary resources. If not, the feasibility study must make clear what new resources the organisation would have to acquire. This will also involve determining whether the hardware and software recommended will operate effectively under the proposed workload and in the proposed environmental conditions. Will new staff be required or will existing staff require training?

The development of a new system involves risks of one kind or another. Every understanding that might be reached could carry the risk of some misunderstanding:

* software companies and their clients often have different vocabularies and consequently they appear to be in perfect agreement until the finished product is supplied.
* management may have unrealistic visions of computer systems.

The feasibility study is where idealism meets reality.

Further issues might include the training of personnel to use the new system, consideration of service contracts, warranty conditions and the establishing of help desk facilities for inexperienced users.

## User surveys

As part of the analysis process, it is important to find out what your end-user requirements are. User research is important because in order to design a successful solution, you need to know who your users are and what they want to achieve with the potential system.

You need to identify the users’ goals, needs and capabilities as well as features that they like/loathe in similar existing systems and new features that they would like to see included in your development.

* The results of your user survey will determine the “End-User Requirements” section of your requirements specification document. This section outlines what the users want to be able to do with the new system.
* You may also encounter ideas and suggestions that you had not thought of and this may provide additional content for the scope and boundaries sections of your requirements specification.
* The end-user group is important because their requirements will help to determine the user-interface design and these requirements will also have an influence on your test plan.

### Determining End-User Requirements for your AH project

You could make a paper-based questionnaire or an electronic one using SurveyMonkey ([https://www.surveymonkey.co.uk](https://www.surveymonkey.co.uk/)).

Examples of user surveys for an AH project here: <https://www.surveymonkey.co.uk/r/3XL2825>

<https://www.surveymonkey.co.uk/r/8B3XZLL>

Note that they lack sufficient detail for AH level (particularly about the user completing the questionnaire e.g. age range; experience level; gender etc. which may have an influence on their responses) and contain spelling mistakes. The second survey is very brief and vague – I doubt much useful information could be collected from it.

### Survey Design (not in AH specification)

Surveys normally consist of two types of questions:

Closed Questions**:** These questions get the quantitative data from users. Responses to closed questions do not tell you about the context, motivation or reason for the response. These questions are normally responded to via a checkbox, radio button or drop-down menu. The data obtained can be easily visualized with the help of graphical representations.

Open Questions**:** These questions provide the qualitative data about a user’s behaviour, actions etc. It tells us how the user thinks about a problem by providing them with an opportunity to explain their reasoning. These questions required a text box to explain the cause. Qualitative responses tend to take a lot longer to analyse but provide rich details.

**HINTS**:

* Try to ask neutral questions and avoid leading questions
* Keep the language and context clear and unambiguous
* Avoid grammatical and spelling errors
* Be clear about your reasons for doing the survey and what the data will be used for
* Respect respondents’ privacy and anonymity. For your AH project, knowledge of your end-users’ ages and genders may be important or it may not. Do not collect unnecessary data.
* If questions are not applicable to some respondents e.g. because of their answer to the previous question, make this clear.
* Use open questions to obtain detailed qualitative responses e.g. if you were asking users if they used a similar product to the one you are going to develop for your AH project and you ask the question “Do you use x?”, the answer would be yes or no. Analysis of the results of this question would result in a quantitative number. If you ask “How would you describe your experience of using x?”, you will get descriptions explaining features they liked/loathed, reasons why they haven’t used it at all, features they would like to see added etc. This will give you excellent qualitative feedback that you can incorporate into your Requirements Specification (e.g. as part of end-user requirements, scope or boundaries)

## Project Planning

If the client agrees to the suggested solution in the Feasibility Study and the results of the User Survey suggests that the proposed solution is one that would meet End-User Requirements, the developer will then carry out a more detailed investigation which will result in the Requirements Specification document.

The Requirements Specification document forms the basis of a legally binding contract between the developer and the client. In very simple terms, it will state that the project will be completed according to the agreed specification within a certain time period and for a certain cost. As soon as the contract is agreed and signed by both the client and the developers, the clock starts ticking.

The contract may also include penalty clauses e.g.

• *If the project takes longer to develop than agreed, then there may be a financial penalty that the developer has to pay;*

• *If the client wants changes made to the specification during the development, then the developers will charge the clients more for the extra work involved in making the changes.*

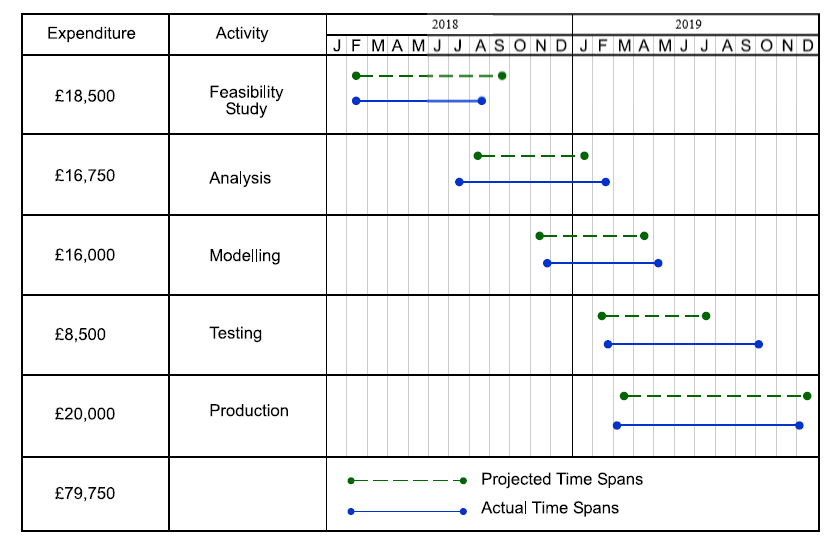
It may also allow for a bonus if the developer completes and delivers the project ahead of time.

It is vital that the project is developed in an orderly manner – this requires careful **planning** to organise the human and physical resources needed to complete the project.

Large project developments will involve the developers being divided into small teams. The project itself will be divided into many small parts, each assigned to a team. **Scheduling** involves making sure that no team is left idle waiting for another team to finish their part.

### Scheduling: Gantt Charts

In many cases of project management, the scheduling component can be aided by means of a **Gantt chart**. A Gantt chart is used to plan and schedule projects involving several **concurrent tasks**.



start and finish times of component parts are graphically represented

The horizontal axis represents the time scale

Advantages of a Gantt chart:

* Shows at a glance the progress of a project.
* Helpful for working out the order in which tasks need to be carried out (the **schedule**).
* Helpful for managing the **dependencies** between tasks (which parts depend on earlier parts).
* Helpful for working out which parts can be completed **simultaneously** (the parallel activities) because they have no dependencies at that particular point in the development.
* Allow you to identify the **resources** (see p13) needed to complete the project, along with the times when these resources will be needed
* Help you work out the quickest possible time in which a project can be completed
* Help you identify the "critical path" for a project. This is the sequence of tasks that must be completed on time if you are to complete the project by a particular date.
* When a project is under way, a Gantt Chart will help you to monitor whether the project is on schedule. If it is not, it will help you to pinpoint the remedial action necessary to put it back on schedule.

Using a Gantt chart makes it easy to identify which activities are **sequential** and which are **parallel**. An essential concept behind project planning (and Critical Path Analysis as well) is that some activities are dependent on other activities being completed first e.g. it is not a good idea to start building a bridge before you have designed it! These dependent activities need to be completed in a sequence, with each stage being completed before the next activity can begin. These dependent activities are called *sequential* or linear.

Other activities are not dependent on completion of any other tasks. These may be done at any time before or after a particular stage is reached. These are nondependent or *parallel* tasks.

Rather than hand-drawing a Gantt chart, you can use a Gantt chart tool instead. This is a piece of software that draws the chart for you e.g. GanttProject (free download from <https://www.ganttproject.biz/>) or Microsoft Project (available as part of Office 365).

**Watch**: Introduction Into GanttProject 2.5 (15:05): <https://youtu.be/5rHCSa5ad34?t=15>

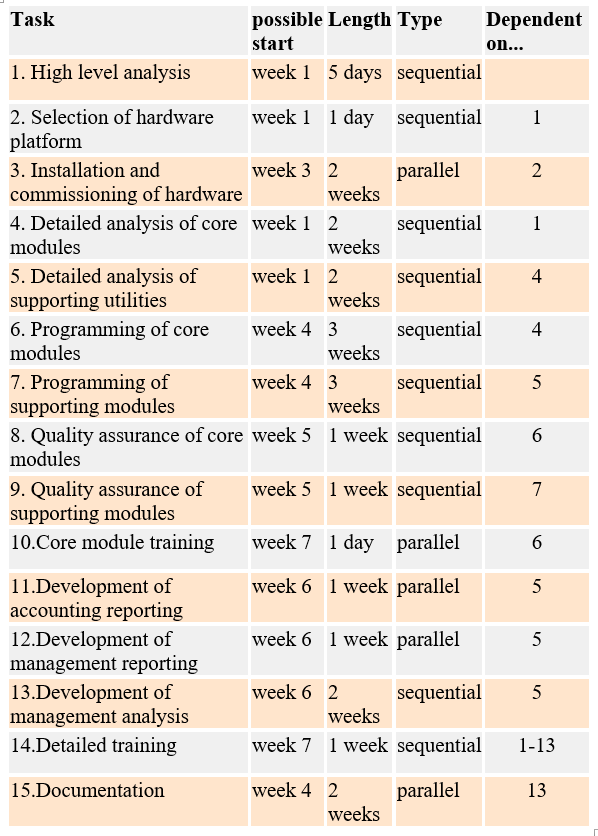
**Download**: GanttProject for Beginners – GanttProject Support:

<https://help.ganttproject.biz/uploads/default/original/1X/ae5fa52935b40a26960ee3c350dbbb311822b542.pdf>

(also available here: X:\Computing\S6 - AH Computing Science\New AH SDD 2019)

For your project:

**1. List all activities in the plan**  
The first step is to list all of the tasks that need to be completed to deliver the project. For each task, show the earliest start date, estimated length of time it will take (and/or estimated due date), and whether it is parallel or sequential. And if tasks are sequential, show which previous stage or stages they depend on.



**2. List resources** that each activity requires e.g. people, equipment, new knowledge etc.

**3. Use a Gantt chart tool** to create your Gantt chart.

TIP: Be prepared for unforeseen problems with your project – try to build some flexibility into your scheduling. Any delay will have a ‘ripple’ effect potentially causing even more delays further down the time line.

The development company’s managers will monitor progress daily, updating and assigning resources to try to keep the project ‘on time’ and ‘on budget’.

TIP: GanttProject has an option for you to export to CSV so that you can analyse your data in a spreadsheet. You can also generate a pdf report. See p8 of the Intro for Beginners pdf.

### Scheduling: Critical Path Analysis (not explicitly in AH but useful)

As with Gantt Charts, Critical Path Analysis (CPA) helps you to plan all tasks that must be completed as part of a project. The critical path is the sequence of linked tasks that has the longest total duration. The critical path has no ‘slack’ and determines the end date of the project.

Critical Path Analysis is an effective and powerful method of assessing:

* What tasks must be carried out.
* Where parallel activity can be performed.
* The shortest time in which you can complete a project.
* Resources needed to execute a project.
* The sequence of activities, scheduling and timings involved.
* Task priorities.
* The most efficient way of shortening time on urgent projects.

The advantage of using CPA over Gantt Charts:

* CPA formally identifies tasks which must be completed on time for the whole project to be completed on time (these are the tasks on the critical path)
* CPA also identifies tasks which can be delayed for a while, if resources need to be redeployed to catch up elsewhere.

The disadvantage of CPA compared to Gantt charts is:

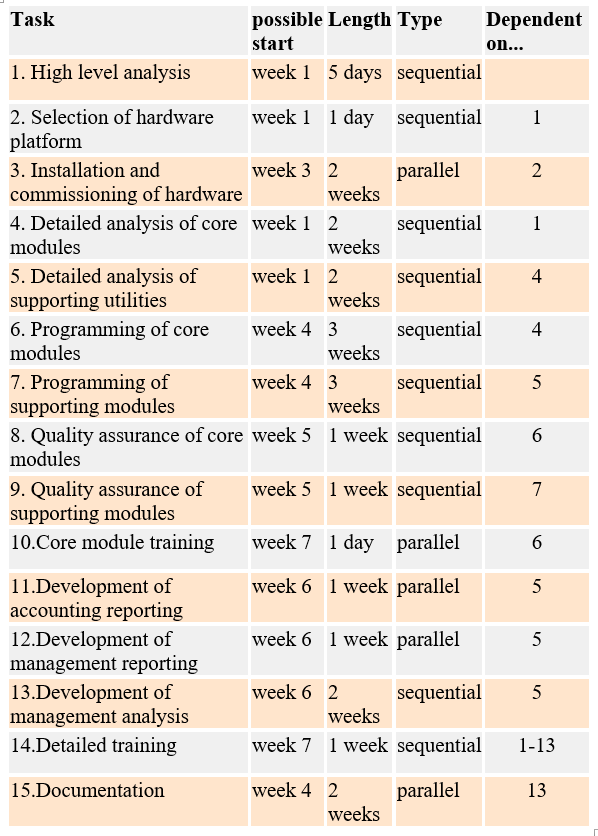
* the relation of tasks to time is not as immediately obvious as with Gantt Charts. This can make them more difficult to understand.

As with Gantt Charts, the essential concept behind Critical Path Analysis is that you cannot start some activities until others are finished. These activities need to be completed in a sequence, with each stage being more-or-less completed before the next stage can begin. These are *sequential* activities.

Other activities are not dependent on completion of any other tasks. You can do these at any time, before or after a particular stage is reached. These are non-dependent or *parallel* tasks.

For your project:

**1. List all activities in the plan**  
The first step is to list all of the tasks that need to be completed to deliver the project. For each task, show the earliest start date, estimated length of time it will take (and/or estimated due date), and whether it is parallel or sequential. And if tasks are sequential, show which previous stage or stages they depend on.

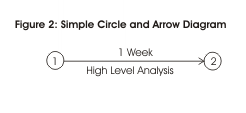


**2. Plot the activities as a circle and arrow diagram**  
Critical Path Analyses are presented using circle and arrow diagrams.

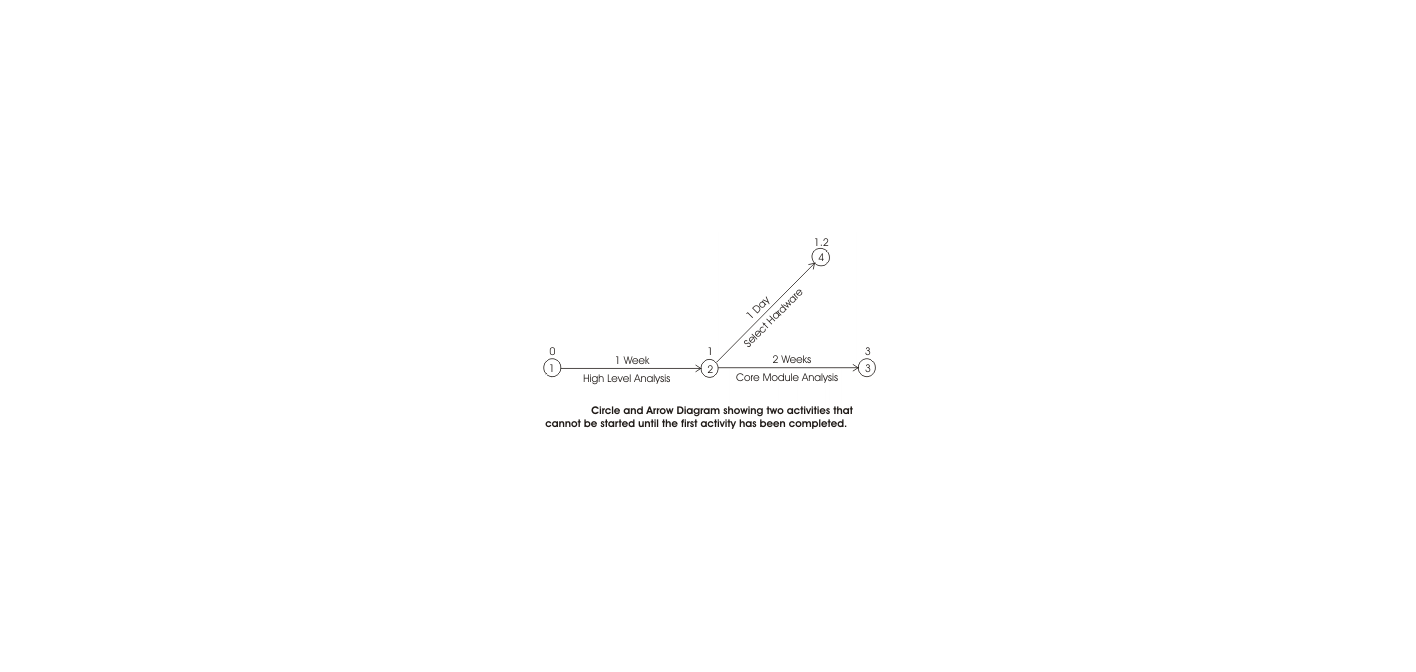
In these, circles show events within the project, such as the start and finish of tasks. Circles are normally numbered to allow you to identify them. An arrow running between two event circles shows the activity needed to complete that task. A description of the task is written underneath the arrow. The length of the task is shown above it. By convention, all arrows run left to right.

An example of a very simple diagram:

This shows the start event (circle 1), and the completion of the 'High Level Analysis' task (circle 2). The arrow between them shows the activity of carrying out the High Level Analysis. This activity should take 1 week.

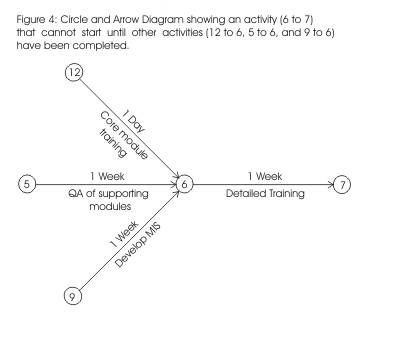


Where one activity cannot start until another has been completed, you start the arrow for the dependent activity at the completion event circle of the previous activity. An example of this:



Here the activities of 'Selecting Hardware' and 'Core Module Analysis' cannot be started until 'High Level Analysis' has been completed.

Another example:



Here, activity 6 to 7 cannot start until the other three activities (12 to 6, 5 to 6 and 9 to 6) have been completed.

TIP: GanttProject has an option to display the critical path but it does not use standard Critical Path Analysis notation.

## **Resources**

Resources include particular people, equipment, additional information (e.g. do you need to ascertain particular facts? Where from?), additional knowledge (e.g. do you need to learn new programming language syntax? Do you need to purchase a new book to do this?) etc. that you require in order to successfully develop your solution.

Examples (not exhaustive!):

* Access to:
  + Individuals
  + Organisations
  + Web resources
* Hardware
* Software
* Data storage requirements
* Access to Web Services e.g.
  + Web server
  + Database server
  + Mail server
* Backup strategy

## Requirements Specification

The requirements specification is how your solution will work rather than just what it will do. It should contain the technical detail lacking in the “purpose” section.

Note: The Requirements Specification document is often the basis of a **legal contract** between the client (customer) and the software company writing the software. It therefore must be **clear, detailed and unambiguous** and **signed by both parties** in order to avoid future litigation.

### Purpose

A high-level description of the problem and the goal, objectives and deliverables of the solution. See p2 for more details.

### End User Requirements

These describe what users expect to be able to do with the system.

This section will be informed by common sense and the results of your User Survey.

Users may want features that you do not have time to develop (see time feasibility) and are outwith the scope of the project. Include these in the boundaries section (see below).

You will already have broadly described your end-users in the purpose section. In the End User Requirements section, you can give more details about your end users, what they expect to be able to do and if different end users require different features, different interfaces, different levels of security etc. These different features, interfaces and security levels should also be stated in the scope section.

### Scope, Boundaries and Constraints (also Assumptions)

* \_ Scope clarifies what the project must cover.
* \_ Boundaries clarify what the project will not cover.
* Constraints are restrictions that apply to the development.

#### Scope

This is a list of the **deliverables** (what the **full development** must contain).

It must be **detailed** and **unambiguous** to avoid future litigation.

Because it is a list of what the development must contain, the scope section goes beyond just describing what the project will accomplish. The scope section should also include *everything* that will be handed over to the client.

For example (in no particular order!):

* What information should be included in the system? (not as detailed as functional requirements section)
* What information is being output by the solution? (not as detailed as functional requirements section)
* What system(s) is it being designed to run on e.g. OS version(s)?
* Are there different features for different users?
* Have you included security? Are there different levels for different users?
* Interface – do different types of user require different interfaces?
* What is being integrated?
* Test plan including test table
* End-user test report (is there a % of acceptable errors?)
* Evaluation report
* Designs (list the designs you will be including e.g. use case diagrams, wireframes, low fidelity protypes, high quality mock-ups, flowcharts, class diagrams, structure diagrams, pseudocode, entity-relationship diagrams, data dictionaries etc. Remember to briefly explain what the particular design is for e.g. wireframe of customer login screen etc.)
* Structured listing (for your AH project)
* Completed program (not required to be submitted to SQA for AH)
* Do you also have to include a user guide (training manual)?
* Time limit (see Project Planning p8)
* Cost

#### **Boundaries**

This sets out in detail the limits of your project. i.e. clarifies explicitly what it will **not** cover.

You (or the results of your user survey) may come up with some ideas that might be reasonable suggestions to include in your project but you do not have time to develop them. You can detail these in the boundaries section as being explicitly excluded.

You should also detail technical boundaries e.g. while the scope should state what OS version(s) you are developing/testing for, boundaries should explicitly state that the system will not run on any other operating system. Also think about file sizes e.g. while your scope should state the maximum file size your system can handle, the boundaries should explicitly state that files larger than your maximum will not be handled.

**Scope and Boundaries Brief Example of Things to Think About**:

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 ...

Sometimes it is also helpful to spell out exactly what will NOT be covered. All these things will define the **boundaries** of the system. 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 any version

on Windows after Windows XP, but not on any other operating system.

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

**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!

#### Constraints

Constraints are restrictions that apply to the development. These restrict the changes made to design decisions during the development. Time, scope and cost are the main constraints of project management; however, depending on the type of development, other constraints may apply, for example:

***Technical constraints***

* knowledge and/or availability of development tools and programming language
* the operating system or platforms that will be used to deliver the working solution
* hardware considerations such as capacity
* non-functional requirements such as performance considerations

***Business constraints***

* schedule and timescales that must be met
* available budget
* composition and makeup of the development team
* software licensing restrictions or requirements

***Further constraints***

economic considerations

political issues

#### Assumptions (not explicitly in AH)

Ideally, all ambiguities and resultant assumptions should be clarified with the client prior to starting the Design stage. This avoids problems with the client later on.

Although not part of the AH syllabus, be aware that it can be necessary to clearly outline any assumptions that have been made prior to design and implementation.

For SQA, you should state any assumptions you have made, particularly if you find the question to be a little vague. This helps to explain your reasoning for your answer.

### Functional Requirements

This section of the Requirements Specification is where you **specify inputs, processes and outputs**. The functional requirements outline what your project should do (the mandatory features) in more technical detail than the Purpose or Scope sections.

It is still relatively top level, however, with more detail produced during the Design stage e.g. you might want to create a top-level structure diagram to show the expected program structure but detailed pseudocode would not be produced until the Design stage.

You should also produce a Use Case diagram clearly showing the actors, use cases and relationships between them (see p??).

Begin by clearly listing the project’s inputs, processes and outputs. For example:

• Inputs

o Each expected input should be listed along with the source of the input (screen, file, etc.).

• Processes

o The principal processes should be outlined e.g.

▪ Opening connection to SQL database

▪ Reading files

▪ Forming and executing SQL queries

▪ Calculations

▪ Sorting

▪ Searching

▪ Interpreting the results from an SQL query

▪ All required validation

▪ Writing files

▪ Closing connection to SQL database

• Outputs

o Each expected output to the user, either as a new or updated screen or webpage, including error messages.

Don’t forget about security. Think about different levels of security (access) for different types of user e.g. customers; site administrators etc.

Integration

Interface, data structure requirements,

test plan – what is to be tested, how it is to be tested, test data & expected results. Test plan should test both functional and non-functional requirements and include criteria for end-user testing.

These four aspects are analysed in terms of:

* **inputs**
* **processes**
* **outputs**
* Security? Think about different levels of security (access) for different types of user e.g. customers; site administrators etc.
* Interface? Will all users of your solution use the same interface or will customers have a different interface from site administrators.

TEST PLAN??

The test plan should test both functional and non-functional requirements and include criteria for end-user testing.

## SQA Examples

\*\*Note that the examples below use the heading “Analysis” rather than Purpose.\*\*

Be aware that both of these examples are very brief and simplistic. You are expected to provide significantly more detail in your Requirements Specification as well as more accompanying documentation in your project (which must therefore be listed in the scope section)! Read this booklet for more information.

### Appendix 1: problem analysis (SDD, DDD and WDD)

**Worked example of a requirements specification (SDD)**

**Analysis**

The purpose of a program is to allow the end user to search for an item on an unsorted list of data. If a match is found, the program will display the row of data for the item.

**Scope**

This development involves creating a modular program. The deliverables include:

* a detailed design of the program structure
* a test plan with a completed test data table
* a working program
* the results of testing
* an evaluation report

**Boundaries**

* the program will read the data (itemID, price, and number in stock) from a sequential file

In my opinion, the way this is written means it is not a boundary but rather belongs in scope. For a boundary, it should be rewritten:

* the program will only read the data from a sequential file, not a random or binary file
* the program will only read the data itemID, price and number in stock NOT <<other data items that will not be read if there are any>>
* assuming the data is accurate, there is no need to implement input validation

**End-user requirements**

End users will expect:

* to enter an itemID while the program is running
* the data corresponding to the itemID to be displayed
* a user interface that is clearly labelled and easy to use for all user types

**Functional requirements**

Functional requirements are defined in terms of the inputs, processes, and outputs listed below. All inputs are imported from a sequential file and all outputs displayed on the screen. The program is activated by double clicking on the file icon and then selecting “Run” from the menu. Each process should be a separate procedure or function that is called from the main program.

**Inputs**

* itemID
* price
* number in stock

**Processes**

* read in data from an external file to a 2D array
* sort the data in order of itemID from low to high
* search the 2D array for the required itemID, based on the end-user input

**Output**

* if a match is found, the data (itemID, price, and number in stock) will correspond to the end-user input
* if no match is found, a suitable message will inform the end user

**Constraints**

The constraints that apply to this development are:

* Live Code, Python, or Visual Basic must be used to develop the program.
* The working program will run on the Windows operating system.
* The work must be completed within 8 hours.

**Worked example of a requirements specification (DDD)**

**Analysis**

GoGoGadgets.com is a company specialising in quirky and unusual gadgets that are available for purchase through its online catalogue.

Before customers can make a purchase, they must first register with the GoGoGadgets website and be allocated a unique customerID.

Customers can browse the product range through an online catalogue. Each item is categorised as one of the following: Toys, Gizmos, Office Distractions, Personal Grooming, and Computer Accessories. All items cost less than £50.

A database is required to store details of customers, items, and orders.

**Scope**

This development involves creating a relational database. The deliverables include:

* a detailed design of the database structure
* a test plan with a completed test data table
* a working database
* the results of testing
* an evaluation report

**Boundaries \*\*need to add comments\*\***

* the database will contain a maximum of 10 000 items
* each item will cost £50 or less
* all items should be categorised as one of the following: Toys, Gizmos, Office Distractions, Personal Grooming, and Computer Accessories
* users must enter a valid e-mail address to register

**End-user requirements**

End users (customers) will expect queries that enable them to:

* register as a user and store their details in the database
* search for items based on the category of the item
* search for items based on the name of an item
* sort items by price (low to high), price (high to low) or rating

End users (administrators) will expect queries that enable them to:

* edit the price of items
* edit customer contact details
* add and remove details of individual items
* remove details of customers from the database
* view details of all orders placed each month

**Functional requirements**

Functional requirements are defined in terms of the inputs, processes and outputs listed below.

**Inputs (customers)**

* register: user e-mail, password, password re-entered, firstName, lastName, address, and postcode:
  + search details: category
  + search details: itemName
* sort details: field (price or rating) and order required (ascending or descending)

**Inputs (administrators)**

* edit item details: itemID and price
* edit customer details: customerID, address, postcode, and e-mail
* add item details: ID, description, category, and price
* delete item details: itemID
* delete customer details: customerID
* monthly orders: month

**Processes**

* auto generate customerID whenever a new customer registers
* queries to:
  + insert records into the Customer and Item tables
  + sort item details in order of price and rating
  + delete a specific customer and an item record from the database
  + edit records in the Customer and Item tables
  + search Item table
  + display details of all orders placed in a particular month

**Output**

* confirmation of successful:
  + insertions
  + deletions
  + edits
* answer tables showing details of:
  + sorted items (sorts)
  + required items (searches)

**Constraints**

The constraints that apply to this development are:

* The Oracle MySQL server must be used to develop the database.
* The working database will run on the Windows operating system.
* The work must be completed within 15 hours.

## UML

### What Is UML?

Unified Modelling Language (**UML**) is a **graphical standardised design language** used to **model a system**. The system may be software or non-software.

UML provides a **clear visual representation** of the system making that system easier to understand and maintain.

Using UML, you can model just about any type of application, running on any type and combination of hardware, operating system, programming language, and network. Built upon fundamental object-oriented concepts including *class* and *operation*, UML is particularly useful in object-oriented software development where it is used to create visual models of software systems.

### Why Use UML?

Large enterprise applications (i.e. ones that execute core business applications and keep a company going) must be more than just a collection of code modules. They must be structuredin a way that enables scalability, security, and robust execution under stressful conditions, and their **structure** (their **architecture**) must be defined clearly enough that maintenance programmers can (quickly!) find and fix a bug that shows up long after the original authors have moved on to other projects.

Clearly defined structure is a good way of dealing with complexity.

Another benefit of structure is that it enables **code reuse**. Design time is the easiest time to structure an application as a collection of self-contained modules or components.

### What Does Modelling Mean? (not explicitly in AH but you need to understand this)

**Modelling** is the designing of software applications before coding i.e. the act of visualising, specifying, constructing and documenting the analysis and design of a system.

Modelling is an essential part of large software projects, and helpful to smaller projects as well. A model plays the analogous role in software development that blueprints and other plans (site maps, elevations, physical models) play in the building of a skyscraper.

Using a model, those responsible for a software development project's success can assure themselves that business functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, extendibility (and other characteristics), ***before* implementation** in code renders changes difficult and expensive to make.

Surveys show that large software projects have a huge probability of failure - modelling is a way to visualize your design and check it against requirements before your team starts to code.

There are currently thirteen types of UML diagrams which are used to represent **dynamic** and **static** views of a system model.

UML diagrams can be divided into two categories:

* **behaviour** diagrams (**dynamic** view)

The dynamic (behaviour) view represents collaboration among objects and changes to internal states of objects through sequence, activity and state machine diagrams.

Behaviour diagrams represent the **functionality** of the software system and emphasise ***what* *must* *happen*** in the system being modelled.

There are seven diagram types representing general types of behaviour. For AH, you only need to know about **Use Case Diagrams**.

* **structure** diagrams (**static** view)

The static (structure) view includes class diagrams and composite structure diagrams, which emphasize static structure of systems using objects, attributes, operations and relations.

Structure diagrams are used in documenting the **architecture** of software systems and emphasise the ***things that must be present*** in the system being modelled.

There are six diagram types representing structural information. For AH, you only need to know about **Class Diagrams**. You will learn about class diagrams in the Design stage booklet.

Watch: UML 2.0

### Purpose of Use Case Diagrams

* Use Case Diagram: describes functionality of a system in terms of actors, goals as use cases and dependencies among the use cases.
* **Use Case Diagram**  
  [Use case diagrams](https://www.smartdraw.com/use-case-diagram/) model the functionality of a system using actors and use cases.
* <https://www.smartdraw.com/use-case-diagram/>

### How to Draw Use Case Diagrams

Watch: UML Use Case Diagram Tutorial (13:23) <https://youtu.be/zid-MVo7M-E>

<http://www.lucidchart.com>

system boundary

actors

use cases

relationships

Example:

### Use Case Questions

# Analysis Questions

**Q4:** In the context of systems development, explain what is meant by a feasibility study

and who carries it out?

**Q5:** Identify **four** kinds of feasibility and describe their differences.

**Q6:** Why is it important that a feasibility study should generate a report?

**Q7:** Explain what is meant by the term *cost-benefit analysis*.

**Q8:** What is the purpose of a Gannt chart?

**Q1:** Explain what is meant by the scope of a software project.

<<insert SQA past paper questions here>>