A-level Computer Science (7517)

Non-exam assessment (NEA) guidance

(Version 2 – June 2016)

Contents

Introduction

The non-exam assessment (NEA) counts towards 20% of the A-level qualification.

The NEA allows students to develop their practical skills in the context of solving a realistic problem or carrying out an investigation. The NEA is intended to be as much a learning experience as a method of assessment; students have the opportunity to work independently on a problem of interest over an extended period, during which they can extend their programming skills and deepen their understanding of computer science.

The most important skill that should be assessed through the NEA is a student's ability to create a programmed solution to a problem or investigation. This is recognised by allocating 42 of the 75 available marks to the technical solution and a lower proportion of marks for supporting documentation to reflect the expectation that reporting of the problem, its analysis, the design of solution or plan of an investigation and testing and evaluation will be concise.

It is recommended that approximately 50 hours of lesson time are allocated to the completion of NEA.

This resource offers some advice on how to tackle the NEA and what to include in the documentation.

You should also refer to the **A-level Computer Science specification** sections relating to NEA

- 4.13 Systematic approach to problem solving
- 4.14 Non-exam assessment – the computing practical project
    - 4.14.3 Marking Criteria
- 5 Scheme of Assessment
- 6 Non-exam assessment administration

For centres currently following the AQA Computing specification (2510), a summary of the most significant differences between the NEA for the new A-level Computer Science specification and the coursework (COMP4) for the outgoing A-level Computing specification is in **Appendix A**.

1 Selecting and approaching a project

1.1 Selecting a project

The project will take a significant amount of time to complete and contributes 20% to the overall A-level grade. It is important that a student selects an appropriate subject for the project that:

- can maintain their interest over a long time period
- is suitably challenging so that it will enable them to fulfil their learning potential
- will enable them to access the full mark range
- can be supported and assessed by the teacher.

It is the student who must decide upon the project subject, but it is expected that the teacher should also be involved. Students can choose between:

- **Solution to a problem**
  The student selects a problem and develops a system to solve it. Typically, the solution would be developed for a third party. There is no requirement for there to be an end user, but having one is likely to be useful. Examples of this type of project include:
  - a simulation, eg of a business or scientific nature, or a well-known problem such as Conway's Game of Life (see https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)
  - a solution to data processing problem for a business, eg stock control, membership systems
  - the solution of an optimisation problem, eg production of a rota, shortest-path problems, route finding
  - a computer game with a dynamically created maze
  - an application of artificial intelligence
  - a control system, operated using a device such as an Arduino board
  - a website with dynamic content, driven by a database back-end
  - an app for a mobile phone or tablet to remotely control a server.

- **Investigation**
  The student selects an area of the subject that they are interested in and conducts an investigation of this area, with the focus being on programming. For an investigation, the student would need a supervisor with some knowledge of the area being investigated. Examples of this type of project include:
  - machine learning algorithms
  - 3D graphics rendering
  - analysis of live data feeds, eg Twitter feeds
  - investigation into the use of neural networks, eg as a noughts and crosses player
  - exploring large datasets for correlations, eg the World Bank's datasets (see http://data.worldbank.org), and creating useful visualisations of these correlations to answer interesting questions
  - scientific investigations, eg where an analytic solution is not possible.

For both types of project, the structure of the documentation required is essentially the same, with minor differences where necessary.

In individual centres the problem chosen by a student to solve or investigate should be sufficiently different to that of other students to avoid the work of one student informing the work of another because they are working on the same problem or investigation. Teachers are required to record on the Candidate Record Form for each student that they have followed this guideline. If in any doubt on whether problems chosen by students have the potential to raise this issue, please contact your AQA advisor.

It is expected that a task that is selected will be of A-level standard. See **Appendix B** for examples of what this means, and how to deal with projects that are not of A-level standard.

Selecting the subject for the project could be done at the end of the first year of the A-level or at the start of the second year. At this point, it may be that some of the techniques required to complete the project (eg SQL, some data structures) have not been taught, so the teacher will need to ensure that the project is realistically achievable.

1.2 Approaching the Project

It is not expected that a student will follow any particular formal software engineering methodology, such as the waterfall model, when working on their project. The different sections of the project report indicate the order that work should be presented for marking, not the order that it must be completed in.

Research
Student's need to research the problem they are going to tackle at the start of the project, which will involve some analysis. It is important that at this stage a well-defined set of measurable objectives are written so that the student and teacher have a common understanding of what the student hopes to achieve and so that the student can gauge their progress.

Critical Path
Once a student has a thorough understanding of the problem, they should be encouraged to identify what the critical path to be followed to produce a solution is, ie what are the key tasks that need to be completed to produce a solution. A student should be encouraged to focus on achieving these before working on more ancillary aspects of the problem such as a user registration and login system.

Designing
It is unlikely to be a good idea for a student to try to design all aspects of a solution before starting to code it. This methodology is outdated and inappropriate, particularly in the context of a project during which a student is likely to be learning new programming techniques as the project develops. Therefore, an agile approach should be adopted where students design, code and test one aspect of their system before moving on to the next, iteratively using experience from testing (and feedback from users where appropriate) to improve their design and coding. Evidence does not need to be gathered to show this development process taking place.

2 The Project Report

The project report should be presented in the following order.

1. Analysis
2. Documented design
3. Technical solution
4. Testing
5. Evaluation

This does not mean that the evidence must be produced in this order. Further detail on what should be included in each section is included below.

The project documentation should be clearly organised, including appropriate titles, page numbering and a contents page so that evidence can be easily found and referred to.

You should also refer to the **NEA marking criteria** in the A-level specification, section 4.14.3

2.1 Analysis (9 marks)

Analysis enables student's to gain a deep understanding of the problem. This in turn allows the student to identify the objectives for the project, giving a clear purpose and direction for their design and solution development.

- **Methods and sources**: Students should be encouraged to use a range of appropriate methods and sources to research and investigate the problem, including websites, existing software, books, interviews, questionnaires, prototyping.

  Relevant and genuine evidence should be presented in the report (most likely in an appendix, but referenced in the main report) but students should be discouraged from generating evidence unnecessarily or artificially.

- **Identifying a third party**: It will be useful if a third party (that is someone other than the student) is involved in the analysis process. The third party might be a potential end-user of the software, such as a friend, relative, employee or teacher or somebody with knowledge, interest or expertise in the problem area. Their role is to support the student in investigating the problem, deciding upon the objectives and to give feedback, particularly at the end of the project.

  For an investigative project, the third party is most likely to be the person who has agreed to act as the student's supervisor. The objectives should be established early on and fixed at a point in time agreed with the third party and the teacher responsible for monitoring the student's project.

- **Further research**: It is accepted that the student's depth of understanding of the problem will grow and develop as the project progresses. They should be encouraged to carry out further relevant research and after consultation with their teacher, refine and reassess their initial objectives as appropriate. The teacher should be the final arbiter and base their judgement on

the initially agreed set of objectives and the scope appropriate for the abilities of the student and the nature of the project.

- **Prototyping and critical path**: Prototyping the critical path at the analysis stage, early in the project development period, is encouraged so that changing objectives later on occurs only in exceptional circumstances, and with the agreement of the teacher. The achievement of the objectives will have an effect on determining the mark awarded for the coding of the project.

  A point in time should be reached when the agreed set of objectives is frozen. This cut-off point should be early in the project development period once the critical path has been established and assessed as achievable.

We recommend that students are given time to carry out prototyping early in the project development period, eg the first term of Year 13, in order to investigate and test key algorithms, data structures and data stores and most importantly demonstrate that the critical path of the problem can be solved in the available time.

This is valuable in enabling students to assess the feasibility and scope of their objectives. However, the initial objectives and the involvement of the third party are important in providing a reference point and structure for the project.

- **Required documentation for 'Analysis'**: this section of the report should include:

  - a clear statement that describes the problem area and the specific problem being solved / investigated
  - an outline of how the problem was researched
  - a statement indicating who the problem is being solved / investigated for
  - background in sufficient detail for a third party to understand the problem being solved / investigated
  - a numbered list of measurable, "appropriate" specific objectives, covering all required functionality of the solution or areas of investigation ('appropriate' means the specific objectives are single purpose and at a level of detail that is without ambiguity)
  - any modelling of the problem that will inform the Design stage, for example a graph / network model of Facebook connections or an E-R model, state diagrams, scientific / mathematical models or formulae, data flow diagrams.

The setting of **well-defined, appropriate objectives** is the most important part of the analysis, as these objectives will eventually be used to assess the success of the project and award marks to the solution produced. See **Appendix C** for examples of objectives.

There are **three mark levels** that can be awarded for 'Analysis'. Section 4.14.3.1 of the A-level specification gives details of the level descriptors.

If a student sets objectives that go beyond the scope of what is required at A-level, these should be indicated as extension objectives by the student, in consultation with the teacher responsible for the student's project. If the objectives were not achieved, this would not prevent a student from achieving the highest available marks for the final developed system.

2.2 Documented design (12 marks)

The importance of design is to assist a student to produce a working, efficient solution to a problem and to enable maintenance of the system at a later date. Evidence for this section of the project report can be produced before, after or during the coding of the system.

- **Sequencing**: It is often useful to complete some elements of the design, such as planning data structures, before coding takes place. This helps students avoid making mistakes which may result in unnecessary work and recoding at a later stage. Other elements of the design that would be useful for maintenance could have their design documented after they have been coded.

  For students, design is also an opportunity to convey the level of technical sophistication of their solution.

  We recognise that *design*, *coding* and *testing* are inherently iterative processes, and not three distinct stages, carried out, one after another. After designing some parts of a system, a student might then go on and code these before other parts of the system are designed in detail. After coding part of a solution it is likely that this part will be tested, which may result in possible recoding and / or redesign.

- **Required documentation for 'Documented design'**: The type of system that a student is developing will determine the aspects of the system that need to be covered in the documented design. It is anticipated that for all systems, a high-level overview of how different parts of the system interact would be useful. This may be a:

  - structure / hierarchy chart
  - a system flowchart
  - a data flow diagram, or object / class diagrams, accompanied by any further explanation that is helpful
  - non-standard diagrams that combine elements of data flow and program control flow are acceptable, as long as the two can be clearly distinguished.

  Students should also document **how the important parts of their system work**. Possible items that might be present in design could be:

  - **Algorithms**: Processing of data should be at the heart of all projects. Students need to show and explain sample algorithm(s) that their project uses. These could be user-defined or standard algorithms, but should be chosen to demonstrate the sophistication of the system and should be key algorithms that are essential to the success of the project. Pseudo-code, Structured English or any other appropriate methodology could be used.
  - **Data structures**: If a project makes use of data structures in memory, these should be explained. Simple structures, such as arrays of records, might only require a short explanation, while a more complex structure, such as a queue, linked list or hash table could be explained in more detail.
  - **File structure and organisation**: If a project stores data directly in files, the structure of the records in these files should be described, together with how the records are organised for access and inter-relationships between files.

- **Database design**: If a project stores data in a database, the structure of the tables should be described and an entity-relationship diagram could be used to show how the tables are related.
- **Queries**: If a project uses a database, samples of the queries that are used, together with explanations should be provided. The samples chosen should illustrate the sophistication of the system.
- **HCI**: In almost all cases, it is expected that there will be on-screen interaction between the student's system and the user. A small number of samples of screen / hard copy, with explanations and reasons should be presented. We recognise that most students will design their HCI on screen, using the features of their chosen programming environment. Therefore, evidence of HCI can be presented as explained screenshots rather than hand-drawn or package-drawn plans.
- **Hardware selection/design**: For most projects, students are unlikely to have a choice of hardware to use, so this section will not need to be addressed. However, some projects are more focussed on the use of hardware, such as Arduino boards. For these projects, it would be appropriate to explain the suitability of the hardware and to present, by any appropriate method, information about the design of the hardware or how the hardware is used by the project.

- **Evidence required**: Students should provide evidence of those things that are relevant to their own project; the list above is not intended as a checklist, on which every item needs to be ticked off. If a project focuses particularly heavily on certain technical areas, then we would expect more evidence for these areas to demonstrate that although other areas may not be covered at all, the project is of suitable difficulty.

  The list above is not intended to be exhaustive; students are free to present other types of evidence that are relevant and useful to their project. For example, a student producing a simulation project might decide to include lifecycle diagrams for entities.

There are **four mark levels** that can be awarded for 'Documented design'. Section 4.14.3.2 of the A-level specification gives details of the level descriptors.

2.3 Technical solution (42 marks)

Credit for the technical solution is awarded based upon two distinct aspects (see illustration on page 11):

- **Completeness of solution (15 marks)** (see below, section 2.3.1)
- **Techniques used (27 marks)** (see below, section 2.3.2)

The **key evidence** that a student needs to provide for this section is their commented program code. Good commenting and the use of self-documenting techniques for programming, such as meaningful identifiers, will make this section easier to assess.

When including the program code, it would be helpful if it were broken down into titled sections and perhaps indexed to aid the identification of evidence.

Students are free to include any other evidence that might highlight the technical quality of the work completed.

2.3.1 Completeness of solution (15 marks)

Marks are awarded based upon how much of the proposed system the student has created. To enable this section to be assessed, it is of vital importance that a student has produced a sound and detailed set of objectives in the analysis and that appropriate testing has been carried out to evidence the achievement of the objectives.

If a student has not produced a suitable set of objectives in the analysis, then this mark should be awarded on the basis of what the assessor believes a reasonable set of objectives for the project would have been. Similarly, if a student has attempted to define the objectives in such a way as to make it unjustifiably easy to achieve the marking criteria, then the assessor should consider what they believe a reasonable set of objectives to be.

There are **three mark levels** that can be awarded for 'Completeness of solution'. Section 4.14.3.3.1 of the A-level specification gives details of the level descriptors.

It is possible that a student might have identified in the analysis extension objectives that go beyond A-level standard. A failure to achieve any of these objectives should not prevent the student from being awarded a mark for having achieved "all or almost all" of the objectives assuming that the objectives which were not classified as being extension objectives have been achieved.

2.3.2 Techniques used (27 marks)

This section assesses the technical skills demonstrated in the solution the student has created.

Analysis of the program code written should be the main route through which the technical skills demonstrated by the student are identified. However, this judgement could be assisted by other evidence:

a) The documented design should include evidence of the data structures and algorithms the student intended to use.
b) Comments should be included in the program code to highlight where techniques have been used.
c) The teacher could look at the program code with a student present and ask the student to explain what they have done and how.
d) The teacher could ask students to write notes to point out where in their program code they have used particular techniques.

(a) to (d) can be used to assist in deciding on a mark, but the mark must be determined by what is seen in the program code as, for example, the student might plan a sophisticated algorithm but not fully implement it, or exaggerate what they have done in a discussion.

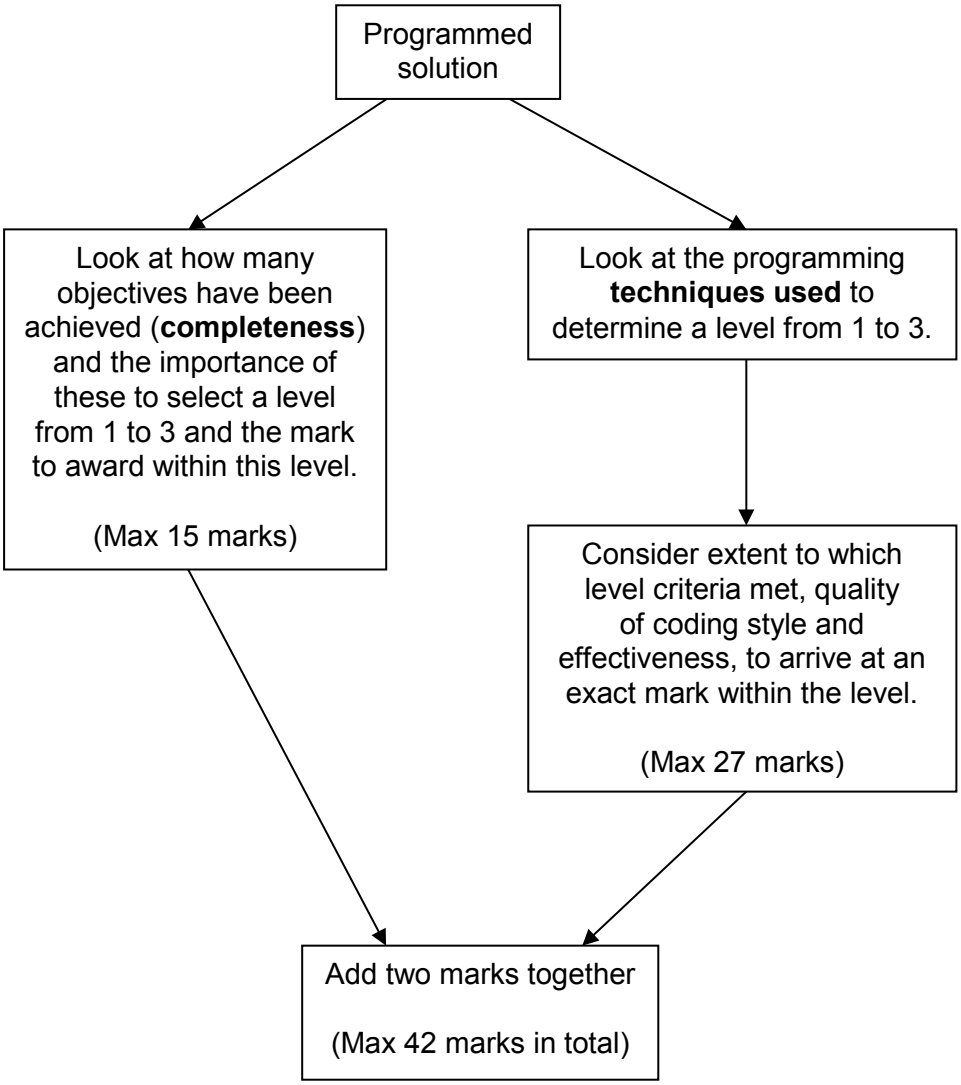
There are **three mark levels** that can be awarded for 'Techniques used'. Sections 4.14.3.3.2 and 4.14.3.4 of the A-level specification provide detailed guidance about the types of technical skills that need to be demonstrated for each level.

Within a level, the specific mark to award is determined by:

- the extent to which the criteria for the mark level have been achieved
- the quality of the coding style that the student has demonstrated
- the effectiveness of the solution.

Section 4.14.3.4.2 of the A-level specification illustrates the qualities that should be looked for when determining the quality of the coding style.

Summary of how to award the mark for the 'Technical solution'



2.4 Testing (8 marks)

Testing demonstrates that the student has, or has not, achieved the objectives identified in the analysis. It is not necessary to provide evidence of testing, or planning to test all aspects of the operation of the system, and the candidate should select and provide evidence of testing those aspects which most clearly demonstrate that the project fulfils its purpose.

There is no simple answer to the question of how many tests need to be carried out. Ideally, the tests completed should show that the system developed has fulfilled its purpose and demonstrates to the assessor the scope of the final system.

Testing does not all need to be carried out on the final version of the system. It is acceptable for testing evidence to be gathered during earlier stages in the development of the system. Informal testing during development has a higher chance of failing and this should not cause a student to fail to document this but rather provides an opportunity to discuss, fix and retest.

- **Investigative type projects**: It is possible that a student might choose an investigative type of problem for their project, where the detailed outcome of runs of the system on input data is unknown. For example, if a simulation is produced, the whole purpose of the system might be to see what happens under certain conditions and testing the core processing functions of the project would involve experimentation.

  With such projects, there might be some tests with known correct outcomes that could be conducted on the parts of the system that can be predicted. The test evidence could also include examples of runs of the system where the outcome is unknown, and so cannot be specified in a test plan other than as a goal, eg the effect of income on life expectancy, together with explanations of what they show. Students attempting such projects should not be penalised for being unable to specify expected outcomes for tests.

Required documentation for 'Testing': Evidence showing that the system works must be presented to enable the assessor to understand how many of the objectives have been achieved when marking the programmed solution section. It is expected that the assessor will have seen the system running when coming to a judgement about this, but the testing evidence can provide further evidence of this and also helps to confirm the assessor's judgements to a moderator.

For each test carried out, it is expected that the student will describe and comment on the outcome of the test. This could be achieved by a test plan or by writing a commentary alongside the screenshots obtained that show test evidence. A suitable explanation of a test would include:

- its purpose if not self-evident
- the test data used
- the expected test outcome
- the actual outcome with a sample of the evidence, for example screen shots of before and after the test, etc, sampled in order to limit volume.

There are **four mark levels** that can be awarded for 'Testing'. Section 4.14.3.5 of the A-level specification gives details of the level descriptors.

2.5 Evaluation (4 marks)

Evaluation allows the student to reflect on the success of the project in meeting the objectives identified in Analysis. The student should also reflect on feedback from the third party and discuss potential improvements and extensions to the solution.

Required documentation for 'Evaluation': This section of the report should include:

- an explanation of how well the objectives have been met and how this was achieved
- a discussion of how the solution could be improved
- analysis of feedback from the third party who was involved at the analysis stage.

A sensible approach would be to copy the objectives from the analysis into the evaluation so that each can be easily commented on. For each objective, the student could judge how effectively it has been met and also comment (if appropriate) on how the solution might be improved in this area.

The student should aim to explain, in outline, how they might go about implementing the possible improvements. It is important that any third party feedback obtained is analysed by the students, not simply included as, for example, an email from the person.

In addition to commenting on the individual objectives, students should also give an overview of the effectiveness of their solution, as it may be that some points would not be covered by commenting on the objectives alone. For example, some ideas for improving the system might be outside of the selected objectives.

The actual feedback obtained from the third party should be included in an appendix. If feedback was also obtained at an earlier stage in the production of the project, for example based upon a prototype, then this could be referenced to in this evaluation so that it could be considered for credit.

There are **four mark levels** that can be awarded for 'Evaluation. Section 4.14.3.6 of the A-level specification gives details of the level descriptors.

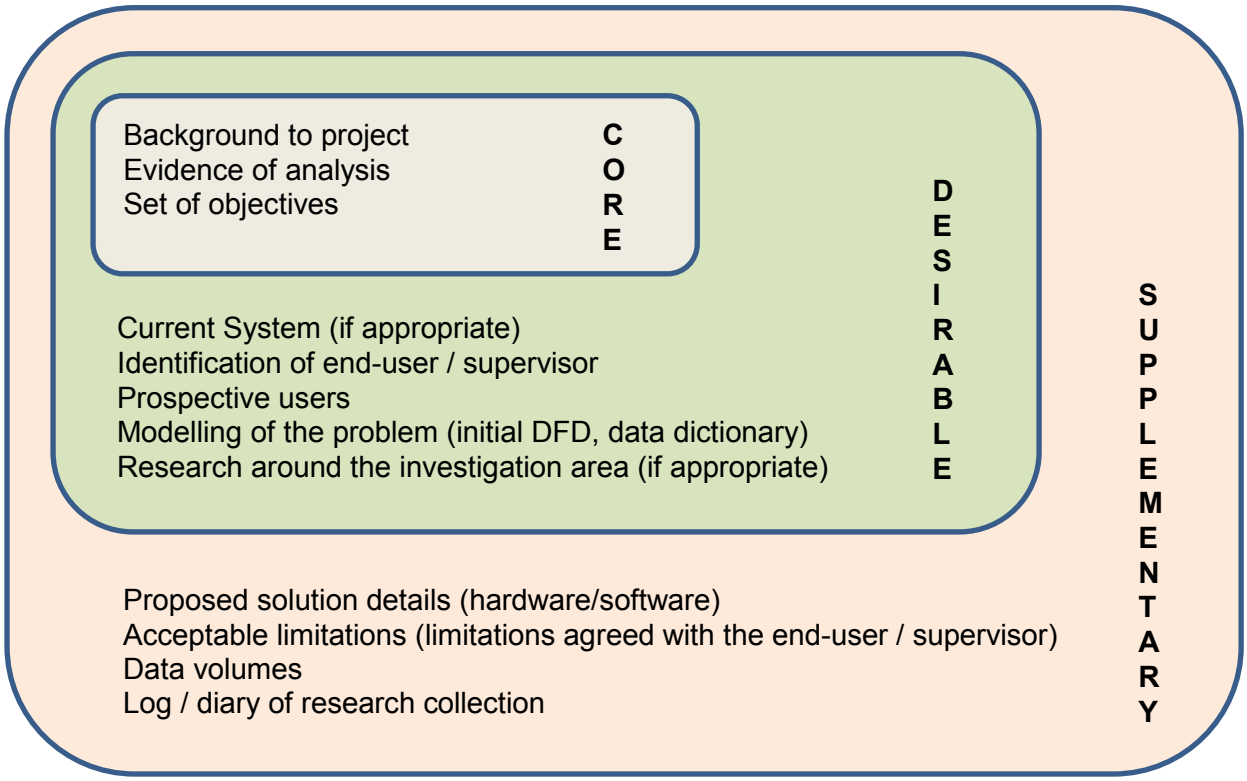
3 Documentation requirements

In this section we provide an overview as to the requirements for each section of the documentation.

Whilst each project is different and therefore a check list of requirements is not appropriate we have tried in this section to identify what we feel are core, desirable and supplementary elements that could appear in each section of a project.

A high scoring project might not have all of the supplementary elements and some will be more appropriate to a problem solving project compared to an investigation.

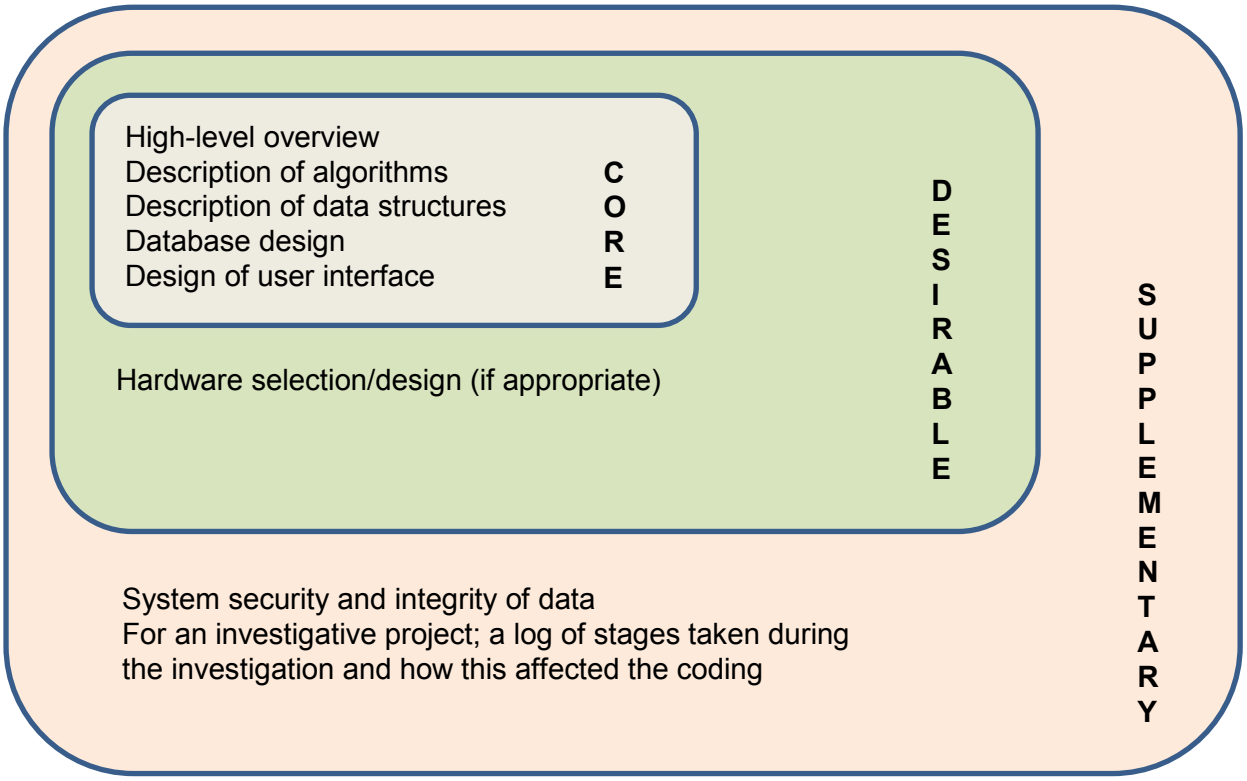
3.1 Analysis



3.2 Documented design

It is important for a student to design how the important parts of their system work.

If, for example, the project is based around simulating the orbits of planets then the equations and algorithms required for calculating these orbits and updating any display would be an important part of the design. A focus on just the user interface with limited discussion of algorithms would not be a fully articulated design.



System security and integrity of data:

If a student has decided to encrypt data in a database or when transmitting data between a client and server you would expect this to be discussed in the design section.

It also might be appropriate for a student to recognise the need for deleting related data when a user is deleted to keep referential integrity.

These sorts of considerations tend to surface more in sophisticated systems where a student also has a good understanding as to issues such as security and integrity of data.

3.3 Technical Solution

The technical solution is usually presented in the form of annotated program listings but these can be provided in a way that makes the marking of the project easier.

The listing could be organised in way that makes it easier to locate certain parts of the solution.

For a web-based project, for example, an initial contents table listing the individual web-pages and the page where the code can be found would be helpful.

In a similar way for a project based around C# a contents table listing the main objects / sub-routines and the page where the code can be found would be helpful.

Overview guide:

An overview guide could contain elements to help the marker including:

- directing the marker to the particularly more sophisticated algorithms
- directing the marker to the code that meets the requirements of particularly challenging objectives.

3.4 Testing

Testing may occur during different parts of the project and when this has happened it is important that a student clearly reflects this in their project. As part of the testing section it may be appropriate to contain a table of tests performed at different stages of the project and a link to any evidence of this.

It would also be appropriate for a student to perform more traditional testing of the solution at the end of coding where the robustness and completeness of the solution are tested.

The requirement is to provide a representative sample of the testing that has happened across the project and not record and evidence every individual test.

3.5 Evaluation

The evaluation should contain the following sections:

- a reflection on the overall effectiveness as to how their final outcome meets the original problem requirements
- an evaluation against each of the objectives set in the analysis section
- independent feedback on their final outcome given by the end-user, supervisor or other third party of the project
- a discussion reflecting on the feedback received and consideration as to how any changes could be made
- a consideration as to how the outcome could be improved if the problem / investigation were to be revisited.

4 Applying the marking scheme

4.1 Analysis

Having read the analysis section of a project the following questions could be considered when marking a project:

- Do you have a clear understanding as to the problem that is to be solved (or investigation to be followed)?
- Has the student undertaken appropriate research into the problem area?
- Is there evidence of dialogue between the student and the end-user / supervisor / third party?
- Are the objectives defined in such a way that the requirements of the problem will actually be met?
- Can you identify the core objectives that will underpin whether the technical solution is complete in terms of the problem requirements?

Objectives

When marking a project consideration must be applied as to whether the objectives set by the student are of an A-level standard. A project that fully meets a set of unsuitable objectives will not be considered complete in terms of the technical solution. A selection of objectives taken from projects are included in **Appendix C**.

4.2 Documented design

Having read the design section of a project the following questions could be considered when marking a project:

- Do you have a clear understanding as to how, in principle, the technical solution will work?
- Having identified the key requirements from the analysis are these sufficiently designed to a good level of detail?
- When comparing to the technical solution do they match up well, eg class methods match those given in the design model?

4.3 Technical Solution

With a greater focus on the technical solution it is important for the marker to have a good understanding as to the sophistication of the solution. One method of applying this is to identify the more challenging objectives and looking for how these have been tackled in the technical solution. A good project would also have clear testing of these objectives which would provide evidence as to the completeness of the project.

Hopefully the marker will have seen the technical solution develop over a period of time and already have an understanding as to the sophistication and quality of the coding.

Completeness of solution
The measure of completeness is to take into account what would be expected of an A-level project. In terms of considering the requirements of a solution a focus should be placed on making sure those that are the most challenging are considered carefully.

Take, for example, a system that is set up to control a stage lighting system using a specific protocol. If this project had 10 objectives which included aspects of user interface design, ease of use and one based around the control of the lights then if the technical solution did not control the lights the project would not be seen as meeting the core requirement.

4.4 Testing

Testing is an opportunity for the student to demonstrate that their solution works and is robust but it can also provide evidence about the sophistication of the solution.

The testing should focus on the more technically challenging parts of the solution.

Demonstrating that a login system works with a variety of usernames and passwords is good at showing robustness. However, if the main focus is on scheduling parent evening slots and this is not tested then the section would not be showing that all of the core requirements have been achieved.

Having read the testing section of a project the following questions could be considered when marking a project:

- Does the section demonstrate that thorough testing has been carried out?
- Have the core requirements been tested?
- Is there evidence that the more technically challenging parts of the project have been tested?

4.5 Evaluation

The evaluation section is an opportunity for the student to reflect on their work and consider how well they have met the requirements of the problem to be solved.

Having read the evaluation section of a project the following questions could be considered when marking a project:

- Has the student given consideration as to how well the outcome meets the initial requirement?
- Has the student reflected on the objectives originally set and considered how their solution measures up to those objectives?
- Has the student obtained independent feedback and reflected upon this?

The evaluation should be an honest appraisal by the student. It might be, for example, that as the project has progressed that another way of solving the solution has become apparent but there was no time to change to this. An honest appraisal would consider this and highlight how this other way of solving the problem might have been of benefit.

Appendix A

Key changes from AQA A-level Computing Project (COMP4)

1. The requirement to judge the overall level of complexity of a project has been removed.

2. The proportion of marks allocated to the technical solution has been **significantly increased** to 42/75 (56%). COMP4 was 20/75 (27%).

3. In addition to the traditional type of project that requires the production of a solution to a problem, there is now an alternative type of investigative project.

4. The documentation requirements have been **significantly reduced**:

    a. The following sections are no longer required:
        i. System Maintenance
        ii. User Guide

    b. The following subsections are no longer a core requirement:
        i. Analysis: Realistic appraisal of the feasibility of potential solutions
        ii. Analysis: Justification of chosen solution
        iii. Design: Identification of storage media
        iv. Design: Description of measures planned for security and integrity of data
        v. Design: Description of measures planned for system security
        vi. Design: Overall test strategy

5. There is greater flexibility over the content of the analysis and design sections, to reflect the wider range of projects that students produce.

6. It has been made explicit that we do **not** expect students to follow a traditional systems lifecycle approach when developing their projects.

Appendix B

Is a project (7517/C) of A-level standard?

If the problem (or investigation) selected for a project is not of A-level standard, mark the project against the standard marking criteria, but adjust the mark awarded downwards by two marking levels (two marks in the case of evaluation) in each section for all but the technical solution. For technical solution, you should have already taken the standard into account for this, by directly applying the criteria. For example, if a student had produced a 'fully or nearly fully articulated design of a real problem describing how solution is to be structured/is structured' this would, for an A-level standard project, achieve a mark in Level Four for Documented Design (10-12 marks). If the problem selected was too simple to be of A-level standard but the same criteria had been fulfilled, shift the mark awarded down by two levels, into Level Two, an award of 4-6 marks. If a downward shift by two levels is not possible, then a mark in the lowest level should be awarded.

The examples below are illustrative of tasks that are **not** of A-level standard.

Noughts and Crosses
An implementation of noughts and crosses, in which the computer validated the legality of moves and checked for a winner, would not be of A-level standard. However, the implementation of some good AI such as the use of a game tree / minimax so that the computer acted as one of the players would turn the game into a project that is of A-level standard. The range of marks that were accessible for the technical solution would depend upon the sophistication of the algorithms used for the AI. A more secure way of ensuring that a project was of A-level standard would be to choose a more sophisticated game which could include more advanced AI, advanced graphics or perhaps network communications.

Rota System
A system that stores information about employee availability and shifts that employees are needed for at a business and matches these two together would be of A-level standard, presuming that it was the program that carried out the matching process and not the user. If the matching was completed by an algorithm them the range of marks available for the technical solution would depend upon the sophistication of the matching algorithm, for example the number of factors taken into account by it (eg staff holidays, qualifications, desired working hours). If the matching was carried out by the user then it is unlikely that the project would be of A-level standard.

Encryption
A program that demonstrates the use of a simple encryption method such as the Caesar Cipher is unlikely to be of A-level standard due to the limited amount of processing done. However, if this were developed into a more complete system, for example with the option to use alternative types of cipher, a graphical display of the Caesar Cipher wheels, the ability to load/save data then this would be of A-level standard but unlikely to achieve a high mark for the technical solution. The incorporation of a code-cracking option that used techniques such as frequency analysis or comparing potential solutions to dictionary text to crack a code would offer access to higher marks for the technical solution, as a result of more sophisticated algorithms being used.

Quiz
A simple quiz program that stored multiple choice questions and answers in a single table or file and scored a user based upon their answers to the questions would not be of A-level standard. The additional of functionality for a user to login and for their scores to be saved, and for questions

to be edited within the program would make the system of A-level standard. The implementation of a more sophisticated technique to analyse responses and tailor future quizzes based upon these has the potential to enable the awarding of a higher mark for the technical solution, as would making the quiz work across a network using TCP/IP so that players could compete against each other.

Appendix C

Example lists of objectives

Project A

These are the full set of objectives from a supermarket product recommendation system:

1. Show confirmation message when changes made to data
2. Users must find it easy to work through the system
3. Pictures should be stored within the database as long binary data
4. The system should be readable to all users
5. The system should not cause any perplexity
6. Random product ID generator should create a random and unique Product ID
7. The system should show any recommendation of the products, depending on the data produced by the algorithm
8. Secure logging in procedure

It is hard from the set of objectives above to actually understand what the project is intended to do. For example, it does not identify that the system is to have a checkout system and recommend products based on what the customer had in their basket. Issues such as calculating the total for a basket and being able to produce a receipt are not covered but were expected requirements from reading the problem background section.

Objectives 1, 2, 4 and 5 are not really objectives to measure the technical solution against for completeness.

Objectives 6 and 7 are beginning to look like suitable objectives but 'the algorithm' was not discussed in the analysis stage of this project and this is where it should have been defined. Due to the student not performing enough analysis into how the recommendation system was to work it can be seen why the student struggled to set a well-defined objective.

Project B

These are a subset of the objectives from a quiz based on puzzles:

3. In order to help the user if they are unable to answer the question, a puzzle must be shown below the question.
    - 3.1 The answer must be calculated based on the randomly generated question and displayed on a grid of 16 squares.
    - 3.2 The first 15 of the aforementioned squares must be saved as JPEG images and the 16^{th} should be replaced with an empty square.
    - 3.3 The order of the 16 squares must then be randomised and presented to the user, below the exam question.
    - 3.4 If it is next to an empty square, the user must be able to move a tile by clicking on it.
    - 3.5 The user must be able to submit an answer, even if the puzzle has not been solved.
4. The questions should be from four different COMP 3 topics: real numbers, regular expressions, reverse Polish notation and finite state machines.
    - 4.1 For the real numbers topic, the user must be shown a mantissa and exponent, and asked to convert them to a fraction.

4.2 The regular expressions questions should describe a string and request for the user to write a regular expression that matches this.
4.3 An input string will be shown to the user, along with a finite state machine and the user must input the output string for the finite state machines topic.
4.4 For the reverse Polish notation questions, a statement in the form of reverse Polish notation will be shown and the user will have to convert it to a denary value.

The objectives above are more detailed and clearly give well defined criteria that will help with the design of the solution, provide measures to test the technical solution against and to finally appraise against in the evaluation.

Activity for defining objectives

An idea for an activity for defining objectives is to take a known application and ask students to come up with a list of objectives for that program.
So, for example, for a simple text editor:

- Allow the user to create, save, load and print simple text documents with no formatting
- Word wrap option that be turned on or off
- Font can be changed but only a single font
- Line and column position of cursor displayed in status bar
- Areas of text can be selected and cut, copied, pasted
- Text can be typed at end of existing text or 'inside' the text, causing existing text to move to the right and down
- Insert/overwrite mode can be toggled
- Documents longer than the screen can be created/handled so scrolling must be implemented
- Search and Replace facility.