

Software engineering is concerned with numerous activities that stretch far beyond the programming and debugging of code. It is a collective term that encompasses a range of processes designed to provide context and validation to development, and to better prepare for and sustain the production of software. In fact, programming ability may be far down the list of requisite skills when looking to recruit software engineers. A software engineer is an expert in identifying the need for software, designing appropriate solutions, and documenting design efforts.

1.1 Expectation of Written Representation at an MSc Level

It goes without saying that academic writing uses formal vocabulary as compared to the vocabulary used in day-day communication. Your aim should be to make your text as clear as possible – to present your ideas clearly and concisely and to avoid ambiguity or redundancy. The tone of academic writing is formal with clear focus on the issue or topic rather than your opinion. Academic writing is defined by conventions rather than specific rules.

As future software engineering leaders, we aspire our students to work to a high professional standard and use relevant formal vocabulary to engage in a critical/analytical discourse as part of their report specification. It also goes without saying that grammar and punctuation need to be impeccable and that you need to be clear on the accuracy of the required vocabulary for this module. If in doubt please refer to the Sommerville textbook or ask the module leader. We wish to bring out the best in our students and we highly value originality and individuality in their written work. We therefore, provide you with all the tools that you will need to model your answers but we do not provide model answers to students to template off. We also do not provide model answers on the online forum in response to questions you may ask of us.

1.2 Submission Report

This section details the report deliverable written to high professional standard. At least that is what aspire from our students. Your report must be a maximum of 15 pages, including diagrams but excluding any references. An allowance of +10% is made for this page limit. The marker will not read beyond this. Your report should be written using font size 12 (or other standard size). Please include appropriate section headings and page numbers in the footer. The core course material is sufficient to pass this assignment well. Reading extra material can be helpful to inspire and refine your solutions. Write from what you know using formal vocabulary and formal definitions, and then move on if needed and if time allows. You are not required to read beyond the core material, but any additional sources of information should be correctly referenced. Your report should include the sections below. The first page needs to be a cover page with you the assignment title SE1 Assignment Report and must not include your name anywhere and the first page is not counted as part of the page count of your written report. The assignment deadline can be found on Engage. You must submit a PDF file to the Engage assignment page by the submission deadline shown above. Submissions received after this deadline will be capped at 40% if received within 5 working days. Any submissions received after 5 working days will be marked at 0%. If you have a valid reason for an extension, you must submit an extension request through your Director of Studies – unit leaders cannot grant extensions. You should leave yourself

time to download your file from Engage and check that you have attached the correct file, with the content that you want to be marked. You are responsible for checking that you are submitting the correct material to the correct assignment.

2 Assignment

You are welcome to ask about a particular process and we will provide a client response where appropriate. However, you do not need to redo any work if you see that someone asks about a process that you have already modelled and made assumptions for. Do feel free to have fun with this assignment and be creative. We will be looking at your models, whether they reflect other parts of your design/requirements, and the accompanying justifications. This is an introductory unit where we are interested in your decisions as a future software engineering leader and understanding of the material. Posting Online in Forums. We encourage you to post all your questions on the forum but we need to remind you that if responding to you means that we are answering a question which is a potential solution; we may refrain a response and our decision stands. Ultimately, we want you to become confident software engineering leaders who are confident with using their tools and engage in a good academic discourse.

2.1 Requirements, 30%

The proposed requirements placed in the booked library brief have several issues. There are deliberate omissions in the scenario so that you have room to be creative in your designs. You could come up with anything else that you think would be a good and justifiable addition to the system requirements for book loans. In a real world software engineering project, there would be some interaction with a client, who may want more or less involvement in the design process of the system. In this assignment, the role of the client is played by members of the teaching team, who have prepared the brief provided and are now passing design decisions to you.

- Identify five issues with the proposed functional/non-functional requirements in the Booking brief. We want you to reflect and identify what these issues might be and write about them; provide justifications for each point. [10%]
- A list of pertinent system and user requirements for Functional Requirement 4 (book loans) and be clear about separating them into user requirements and system requirements. [20%].
- Do not include all system/user requirements, just new ones that you have created. In your requirements, you need to demonstrate that you have provided consistent requirements that show a clear and complete distinction between what the system shall accomplish (requirement) and how a feature is implemented(design) and a clear distinction between the system requirements and user requirements.

You have to show that you have done dependency checks between requirements; you have done conflict check between requirements and demonstrate how you have resolved the conflicts between requirements and how you have demonstrated measurability in your requirements. You will be creating the details of these requirements yourself, so the book loaning and any other associated processes are up to you. You are not expected to modify the existing set of requirements, but can indicate if you think your requirements will affect existing ones.

Measurability can be demonstrated by your requirements in the following ways; this list is not exhaustive: Complete, Traceable (using a traceability matrix is fine), Consistent, identify the constraints placed on the requirement(s) and as this list is not exhaustive so we will be happy to accept any other method you have chosen to demonstrate how you will measure your requirements. We do not expect you to divert to non-functional requirements.

These are some ways in which you can show that you have worked to measure your requirements. You can show this with examples as part of your booked brief.

This list is not exhaustive and you can bring in further ways of measuring your requirements.

- o Completeness – Each requirement should be complete. This can be demonstrated by the fact that it clearly and distinctively translates into a relevant and distinct system feature and there is a clear indication that all its sub-requirements are exhibited by a clear dependency structure and any conflicts are resolved. Again, make sure that the students do not use attributes in place of sub-requirement.
- o Traceable – This does not require any skill outside of the material stated, but traceability needs to be demonstrated through either linkage between requirements using a clear numbering system or traceability matrix or even a traceability tree diagram all which are focused on showing the linkages between all the requirements for a given module. Here is a link from Sommerville that shows how to construct a **traceability matrix**. A traceability matrix can be designed to view conflicts, check for dependencies, check for "resolved conflicts". You can also construct a tree diagram if you want but choosing one representation approach over another does not affect your grading as long as whichever approach you have shown is clearly explained.
- o Consistency Check – Requirements need to be consistent throughout; with each other; consistency can be checked by examining each requirement in relation to each other for completeness and compatibility. For a given requirement, are all the sub requirements also system requirements? and same in the case of user requirements.
- o Testable – Common sense checking on the student's part to demonstrate that the stated requirement is specific and could be easily mapped to a system feature, unambiguous and quantitative where possible.
- o Constraints – During the requirements engineering phase, students may identify constraints that they identify could be placed on the requirement and it is essential that these are not design constraints; these could also be explained via pre-condition/post-condition checks/ source-destination/alternative conditions.

o Pertinent: they do not include unnecessary details/ introduce details outside the scope of the system specification in the requirement without clear assumptions/ adding non-functional aspects into the functional requirement. Makes logical sense.

Below are some examples, but we are also open to other written representations that you choose to use to explain your requirements.

1. You can present them as follows

(Precondition - PRC, Post Condition - POC, Conflict - CF, Dependency - Dx)

System

Fx1.x State your high-level functional requirement

Fx1.a State your sub-level functional requirement (PRC-None, POC- None, CF - Fx.b, Dx - Fx.q)

Continue this way, you can choose to write your own notation for Precondition, Post Condition Conflict, Dependency or you can choose to use the notations of the ones that I have presented to you here.

Continue this way with the user requirements, using the same notations.

2. Another way to present your requirements, you will be able to use a table.

Requirement (Functional/Non-functional)	Pre-condition	Post-Condition	Dependencies	Conflict Resolution
System Requirement Fx.2.1 -write the whole requirement	Fx.1.1	Fx2.2.	Fx2.1- 2.y	So here you need to explain what you did to resolve the conflict you spotted between the two requirements. Simply stating the requirement conflict is not enough – what did you do to “resolve” the conflict – what did you do? You need to show your decision making.
Fx2.2- write the whole requirement	

User Requirement Ux1.2 - write the whole requirement	State which requirement or feature is a pre-condition. This could be a user or system requirement	State which requirement or feature is a post-condition. This could be a user or system requirement	State what is the dependency here.	Again, explain what conflict was resolved and if there isn't a conflict then say so and why is there no conflict for this requirement.
---	---	--	------------------------------------	--

Conflict Checks - These are checks that ensure that given two requirements are in not in conflict with each other. Lets take an example, say you have a system requirement as follows.

Fx.y The system should allow for the available parking slots to be displayed on the parking display.

Fx.p The system should allow for a chosen slot to be booked for reservation or immediate use.

Now this is a situation where 2 system requirements can be in conflict with each other, so the slot which is being reserved must NOT appear as part of the available slots for booking parking reservations. This is a case where a conflict will arise if a new user is trying to book the same slot that is part of the ongoing booking. Now, this is something you will need to think if such a conflict arises in your booked brief or not?

Dependency Checks - Now using the same scenario above, we can make sure that we run a dependency check where Fx.y is dependent on checking Fx.p in order for a booking to proceed forward. So Fx.y must not include the slot that is part of the ongoing booking, hence Fx.y is part of dependency check as part of the checks you will be doing.

This is an example of how you can demonstrate conflict and dependency checks, you can take some of the requirements and demonstrate this through an example and written explanation. You are not expected to do this for all requirements but only some. This is an example of demonstration.

2.2 Architectural Design, 30%

This section must include

- An architectural design for your system using suitable notation.
- A written justification which includes a comparison between 1-2 other architectures. It is important that you understand how to write justifications; simply describing your viewpoint is not enough and we encourage our students to write a critical analytical argument for the choice of the architectures and why you have selected this one.
- A clear list of non-functional requirements which are supported by clear justifications; should you choose to discuss/include these when you compare different architecture styles and their suitability.

You may want to refer to the architectural patterns lesson for this. You also need to read the study skills required for the SE1 unit on the online forum post so that you can get an understanding of the expectation of written representation at an MSc Level. To show diagrams of your architecture models, you should use suitable abstractions and connection to requirements and accurate notations. Your justifications should include selected design, comparisons against alternatives.

Note: Since how people would approach varies, we have given a broad choice here. Some students would like list NEW non-functional requirements and use these as the basis in their architecture, as they feel it pertinent to the architecture style, some may not choose to use this approach. How you choose to use the non-functional requirements is your application choice.

When you attempt the question on comparing and contrasting system architectures, we are happy for you use the following architecture characteristics for comparing and contrasting two system architecture styles. These are as follows.

1. Availability
2. Reliability
3. Testability
4. Scalability
5. Security
6. Agility
7. Fault Tolerance
8. Elasticity
9. Recoverability
10. Performance
11. Deployment
12. Learnability

2.3 System Models, 30%

This section should include:

- A set of system models using suitable notation. It will not be possible to model the entire system in a 15 page document, so be selective and model a few key functionalities. We suggest that your system models should include: A model showing the context of the system (business process model or context model).

- 1) A use case diagram(s).

- 2) 1-2 sequence diagrams modelling some key functionality in your system. Note: you are not expected to create models for the entire system.

- 3) A class diagram.

- The System models should include suitable and accurate notations for your choice of modelling, use of suitable abstractions and connection to requirements. A minimal description for each system model should be included. This should describe what is being modeled and explain why it is being modeled.
- The justifications of your system models must include a balanced critique of the aforementioned system models.

2.4 Presentation Report, 10%

The overall quality of your presentation will be graded for professional presentation, coherence, grammar, punctuation and the quality of your written report