2.2 Coursework I: The Engineering Design Review

When joining an organisation, it is unlikely that you will start an entirely new piece of software from scratch. Instead, you will be contributing to an existing software system. Depending on the organisation you work for, the process by which work is planned and allocated to you may vary considerably. However, it is likely that the process is primarily driven by business needs in one way or another. Some examples of common processes include:

- Your organisation's leadership decide what to work on, based on their thoughts on what will allow the organisation to make (more) money. Your management chain may then divide the necessary work up and allocate some to you.
- The leadership team has identified broad priorities for the organisation which product managers work to design the product around. Product managers will then coordinate with engineering managers to plan how the product they designed can be engineered, and some of the resulting work is then allocated to you.
- Some organisations may give total freedom to engineers to work on features that they consider to be important.

Regardless of the process used, software engineers like you will be required to come up with solutions for the features that are a business priority. For example, you may be told that some feature is deemed necessary and that you should work on it, but it may be up to you to figure out how exactly it should work and integrate with other parts of the software.

Where there is a design space – in other words, there are multiple different options for how a given feature could be implemented – it makes sense to describe the approach that you think is best in a document for an Engineering Design Review (EDR) and compare it with other alternatives that you considered. The EDR is an opportunity for other engineers on your team or other teams who would be affected by your implementation to give you feedback on the proposed approach.

Sometimes, EDRs can also be a means by which engineers can propose some feature that they think is worth working on. In that case, it is likely that engineering and product managers will also review the document to decide whether the feature is indeed worth allocating time for and fits in with the overall product direction.

2.2.1 Task

Choose one of the organisations introduced in Section 2.1. You will stick with this organisation for all assessments in this module.

Your task for the first coursework is to write an Engineering Design Review (EDR) for one project in that organisation. See Section 2.2.3 for guidance on projects.

2.2.2 Structure & Assessment

Your EDR should be a document of roughly 5 pages that you submit as a PDF. The structure of your EDR should be as outlined below.

For each section in the outlined structure below, a number in brackets indicates roughly how much of your EDR should be dedicated to that section, as a percentage. Your EDR will be marked on the *technical content* as well as the *quality of the technical writing*. In marking each part of your EDR, equal weight is given to the quality of the technical content and how well it is communicated to the intended audiences. For your feedback, you will get a mark for each section, as well as overall suggestions for how the EDR could be improved.

Introduction (20%) Establish what problem your EDR is addressing, how the EDR relates to existing systems in the organisation, and why solving the problem should matter to the organisation.

To write this part, you should read through your chosen organisation's description in Section 2.1 and think about the goals of your chosen organisation as well as how the products described may be broken down further into smaller systems. You may make any reasonable, technical assumptions about how the products work internally. Indeed, part of your work here is to think about how the described products may work.

Goals and non-goals (2%) Using quantifiable metrics, list what the design proposed by your EDR aims to accomplish and what it explicitly does not aim to accomplish. In particular, think about what "success" means for the project and how it can be measured.

Design overview (30%) Describe your chosen design in sufficient technical detail that other engineers can decide whether the design is sensible and the best option for the problem at hand. You do not need to describe every little aspect of the implementation, but just enough to communicate how you suggest solving the problem at hand in a way that other software engineers can understand and review.

Alternatives (20%) Give a brief overview of other design options that you considered. Discuss their advantages and disadvantages compared to the design you chose. Most problems have many possible solutions. You do not need to describe every possible solution, just some other ones that could reasonably chosen but you rule out for the reasons given. If you truly think that there is no alternative to the design you proposed, you should explain why that is the case.

Milestones (10%) Propose a plan for how work on the design will proceed if the EDR is accepted (in other words, if other engineers think it is a reasonable design and managers allocate time to work on it). Typically, most projects are broken down into a few, numbered milestones that allow for evaluation at each step once they have been reached to determine whether work should proceed on subsequent milestones. This should cover different phases of implementation, testing, and deployment.

Dependencies (4%) List which other teams in your organisation are affected by the changes proposed in your design. Since your chosen organisation does not really exist, you should make reasonable assumptions about how your chosen organisation may be organised into different engineering teams that are responsible for different aspects of the products.

Cost (2%) Ignoring staffing costs for the project, would the proposed design affect the upkeep of the system for the business or make the product more expensive? For example, are you anticipating changed server hardware requirements or cloud resources?

Privacy and security concerns (5%) Think about whether your design involves collecting any new personal data, or processes existing personal data in a new way that users may not have consented to. If so, discuss this here, how the data will be handled, and what permissions must be obtained from users. Additionally, discuss any security concerns that may apply to the design.

Risks (5%) In a table, describe the main risks of your design, the impact that they would have, and any mitigations that already exist or you propose.

Supporting material (2%) Link to any relevant material, such as documentation for tools, frameworks, etc. that you referred to while writing the EDR or that reviewers may wish to refer to in addition to what you wrote.

2.2.3 Suggested projects

For each of the available organisations, there are suggested projects that you could tackle with your EDR. You may also think of your own project to write an EDR for, but should ensure that it is comparable in scope to the suggestions. In particular, note that for the second coursework you will, as a group, *prototype* one of your group members' EDRs. If in doubt about whether your own idea is suitable, feel free to ask.

Student Smart Homes (SSH)

1. In some student houses, students use grocery delivery services to have groceries delivered to their house. To save on delivery fees, all participating students in the house may place orders together. This project is to extend SSH Console Table and SSH App with the ability to add items to a shared order for the next delivery. Student Smart Homes has partnered with several supermarkets that provide information about available products along with their current prices that can be searched and added to the order. Prices may change over time as promotions come and go or the base price of a product changes. The students should be able to view the items in the order at any time, including who added what, what the total cost is, and the total cost of all items for each individual member of the house is.

2. The SSH Camera can provide detailed information about the contents of the fridge, including quantities. This project is to extend SSH Cloud with a recipe suggestion function that, using the data about available ingredients from an SSH Camera, and a database of recipes, suggests possible recipes using those ingredients. It may also suggest recipes for which most ingredients are available and only some need to be purchased.

3. The SSH Camera can also provide information about who is at home at any given point. Using occupancy trends over time that are available from data collected by the SSH Camera, this project is about adding a feature to the SSH ecosystem that automatically suggests times during the week which may be convenient for a given student to clean their dishes or take out the bins because they are typically at home. Conversely, the system may suggest making use of the TV or a gaming console because all or most other members of the household are typically away.

Higher Education Software Solutions Plc

1. To produce good feedback for students, it is helpful to have *custom feedback forms* for assessments that markers can fill in. Currently, markers are only given two inputs per student submission in CSRS: a numeric input for the overall mark, and a free-form text field for feedback. This project is to design a feature that allows module organisers to specify custom feedback forms,

which may comprise different sections corresponding to different aspects of the assignment:

- Instructions for each section.
- For each component of the assignment, an input for a component mark.
- Multiple free-form text fields for different aspects of the feedback.
- The ability to label submissions with tags (e.g. "1st SSH project") to allow markers to quickly filter all of their assigned submissions by a given tag.

Module organisers should have a convenient way to specify such custom feedback forms for each assignment, markers should then see the form for each submission that they can fill in, the data should be savable for each submission, and students should later see the feedback in the form.

2. For assessments in Computer Science (and some other sciences), there are steps which can be automated once a student has submitted some work. For example, for a programming assignment, compiling the code, running tests, and comparing the student's code to the skeleton code to see what changes have been made. Currently, markers for such assignments need to download the submission from CSRS and perform all of these steps manually on their own machines. This project is to implement a feature which lets module organisers configure a sequence of steps that should be performed automatically once a student submits something. The results should then be made available to markers alongside the actual submission.

3. Student numbers have consistently been increasing for many years. This means that marking coursework for large modules typically requires several people to mark all submissions. For a module organiser to ensure that each marker is consistent is difficult, even with detailed mark schemes. This project is about trying to *detect differences in marks across markers*, and *suggest mitigations*. This may take place once all submissions have been marked and all students' marks are available along with the information about which marker marked which submission. Alternatively, it could take place sooner once there is enough data available. The detections made by this system may indicate that one or more markers are harsher than others, or more generous, allowing the module organiser to intervene. If useful, you may assume that you also have access to students' marks from previous assignments across different modules and years, as well as all other records about each student that are held by a university.