The University of Melbourne

SWEN90016: Software Processes and Management

Sample Exam

This sample exam is handed out for two reasons: 1) to give you a taste of the style of questions that may appear on the exam; and 2) to give you a practice run. It is by no means meant to be taken as a hint to the questions that will be on the exam, nor should it be used as a study guide. This document should be used in conjunction with the subject notes, the lectures, and the tutorials.

This exam, as well as the final exam, contains 100 marks. The final exam is worth 50% of your final mark, so each mark on the exam is worth 0.5%. The number of marks assigned to each question should be an indicator of the amount of time you should spend on each question; that is, 1 mark = 1 minute. The recommended strategy is to spend 1 minute per mark, which for a 2 hour exam, gives you 20 minutes at the end to check your answers.

Part A – Short Answer Questions

Question 1 [3 marks]

What is the purpose of the *Scrum Sprint Review*, who is invited and when is it held during the Scrum life cycle?

Question 2 [3 marks]

What are the roles in a *Scrum Team*? What are the responsibilities of each role?

Question 3 [4 marks]

Describe the term *burndown* in the Scrum context.

Question 4 [5 marks]

A life-cycle model in software engineering is a generic sequence of activities that can be used to achieve a goal. Discuss the main factors that should be taken into consideration when choosing a life-cycle model for a project.

Question 5 [5 marks]

Influence and *power* are used by project managers to motivate people. Compare and contrast the role of influence and power in motivating people.

Question 6 [5 marks]

Describe what risk is in the context of software engineering product development and briefly describe the three key, iterative processes of risk management.

Question 7 [5 marks]

Compare and contrast the fields of computer science and software engineering. What are their similarities and differences?

Question 8 [10 marks]

Below are *three* (**3**) propositions.

You must choose exactly *two* (**2**) of these propositions and clearly state whether or not you agree with the proposition and **why**!

To justify your choice, you will need to write one (**1**) or two (**2**) short paragraphs. Your justification should refer to the relevant theory whenever possible and you should argue whether or not the proposition is supported by your understanding of software engineering theory and principles.

> **Proposition 1:** The classical formal lifecycles of software engineering, such as the waterfall or increment models, are for stuffy old software engineering dinosaurs. The world has moved on, and agile practices are the only sensible way moving forward, because they focus on coding.

> **Proposition 2:** In risk management, all we really need is an estimate of the probability of a risky event occurring. The impact is not relevant because a low probability event is not worth considering.

> **Proposition 3:** The only technique really needed for requirements elicitation are interviews.

Part B – Long Answer Questions

Table 1 contains a work breakdown and dependency information for the requirements phase of a project. The process used is an incremental process, in which two phases of elicitation-analysis-model have been used to produce a validation requirements specification.

Work Breakdown	**Dependencies**	**Most likely**	**Optimistic**	**Pessimistic**	**Expected**
Requirements					
1 Iteration 1					
1.1 Elicitation	–	2	1	3	2
1.2 Analysis	1.1	3	2	4	3
1.3 Model	1.2	3	2	4	3
2 Iteration 2					
2.1 Elicitation	1.1	3	2	4	3
2.2 Analysis	2.1	3	2	4	3
2.3 Model	2.2	4	3	5	4
3 Specification	1.3, 2.3	5	4	6	4
4 Validation	3	4	3	5	4

Table 1: A work breakdown and dependency information for the requirements phase of a project.

Question 9 [15 marks]

(i) Using the information in Table 1, construct a complete PERT chart for the phase.

(ii) Using the durations for each task, what is the **critical path** for the project?

Question 10 [15 marks]

Ejiogu proposes the following six attributes that make a metric useful:

1. **Simple and computable** — calculating (and learning how to calculate a metric) should be straightforward.
2. **Empirically and intuitively persuasive** — a metric should appear valid when proposed, and be in line with the expectations and experience of software engineers.
3. **Consistent and objective** — two different people applying the same metric to the same artifact should arrive at the same answer.
4. **Use of consistent units** — the computation should not lead to combinations of units; e.g. multiplying people on the project by lines of code is not intuitively persuasive.
5. **Programming language independent** — metrics should not be based on the syntax of programming languages, as these differ so much that different languages will end up with wildly different results.
6. **Useful for providing feedback** — a metric should be useful for providing feedback to improve the artifact.

Cyclomatic complexity and function-point analysis are both metrics for measuring the size/complexity of software-related artifacts.

Use Ejiogu's six attributes to compare and contrast these two metrics. If you identify any differences between the two metrics relating to a particular attribute, explain why you believe them to be different.

Questions 11 and 12 refer to the project description below.

Consider a case in which you are the project manager of a team of software engineers in the military. Your team has been given a new project.

The Project

This project requires your team to develop a new software system that replaces an existing legacy system. The existing system has been in operation for over 30 years, and the requirements for the system have remained unchanged since shortly after its initial deployment. The current system is stable, however, it is implemented in the FORTRAN programming language and runs only on ageing hardware. The new system is being commissioned to run on modern hardware in order to improve efficiency. At the same time, the software will be written in a more modern programming language.

Project Inputs

Your team has been provided with a complete requirements specification that has been validated, and will not undergo any changes. Your team has also been provided with the acceptance tests for the previous application.

The previous designs and source code were stored on tapes that are no longer readable, so are not available to the team.

In addition, your team has been provided with the following COCOMO II early-design phase estimate:

Effort	26 person-months
Time	7 months
People	3.7 people

As a result of these estimates, a deadline of 7 months has been imposed on the team.

You have been give a team of four software engineers; the software engineers have expertise all phases of the software development lifecycle, but one of them is known to be a *Free Rider*.

Question 11 **[15 marks]**

Design a software development lifecycle model for the project. Justify your design with respect to the following:

- The key characteristics and goals for this project, and how your lifecycle models best fits the goals of the project.
- The processes you would put in place to achieve the goals of each phase of your chosen lifecycle process.

Question 12 **[15 marks]**

What team structure would you use for the project? Justify your stance by arguing how your team structure suits the project goals, risks, and characteristics, and how it fits within the lifecycle process.