Office Use Only

MONASH University

Semester Two 2018
Examination Period

Faculty of Information Technology

EXAM CODES:	**FIT2101**
TITLE OF PAPER:	**SOFTWARE ENGINEERING PROCESS AND MANAGEMENT - PAPER 1**
EXAM DURATION:	2 hours writing time
READING TIME:	10 minutes

Marking Scheme

Question 1. (2 + 3 + 2 + 5 + 4 = 16 marks)

*This question is about **requirements**.*

User stories in Agile can be evaluated by the INVEST criteria. The V in INVEST stands for "Valuable".

a) In your own words, what does "valuable" mean in this context? **(2 marks)**
Does something that is *visibly useful* to the user.

b) What risks arise if user stories are not valuable? Write about a paragraph. **(3 marks)**
Max 1.5 mark per clearly-articulated, reasonable risk – if students give more than two, use the two best. Max 1 for any risk that is vague or unclear; 0.5 for risks that seem unrealistic.

Examples:
Team may waste time working on features that are not wanted, which risks running over time.
If the project doesn't keep moving towards completion, it may run over time.
May cause (or be a symptom of) scope creep with all its attendant risks

c) How can a team verify that their stories are valuable? Write a couple of sentences. **(2 marks)**
Check them with the customer or Product Owner. Not really possible to do it any other way! (note: for commercial software that is to be sold to the public, the "customer" will be represented by the product manager – no penalty for omitting this point though.) Deduct 0.5 – 1 mark if answer seems to be on the right track but is unclear.

d) What is the difference between a *user story*, a *task*, and a *product backlog item* (*PBI*)? Do tasks and PBIs also need to be "valuable"? Why/why not? Write about half a page. **(5 marks)**
A PBI can be a user story, or anything else that the team chooses to manage by putting it in the product backlog – e.g. spike story, developer story, maintenance task, etc. A task is typically an action that forms part of a PBI, and it is put into the sprint backlog when PBIs are broken down and assigned during sprint planning. Tasks and PBIs don't need to provide customer/user value. They capture things that the team needs to do in order to ship a quality project but might not be visible to the customer, such as analysis, research, refactoring, and infrastructure maintenance

2 marks for clearly distinguishing these three items.
* 2: all three items are distinguished or clearly defined
* 1: two of the three items have been distinguished or clearly defined
* 0: no significant attempt

1 mark for stating why/why not (if they've said "yes", pay this mark only if supported by a very compelling argument! Or allow consequential mark if they've misunderstood "valuable" in a suitable way: see their response to part a of this question.)

2 marks for evidence
* 2: solid argument, well-evidenced, correct understanding
* 1: reasonable attempt made at argument but it is flawed or facts are not correct
* 0: no evidence provided

e) Here is a nonfunctional requirement expressed as a user story:

"As a customer, I want the system to be extremely secure and highly responsive so that I can perform transactions very quickly without worrying that my bank account information will be compromised."

What is the problem with this requirement as currently written? Suggest an alternative version that does not have this problem. **(4 marks)**

Two major problems here: it's vague ("extremely secure", "highly responsive") and it's a compound requirement that covers more than one thing: speed and security.

Break up into two requirements: one addressing speed ("As a customer, I want the system to respond within 0.5 seconds after I have entered a transaction so that I can perform transactions very quickly") and one addressing security ("As a customer, I want the system to <*any plausible security-related story can go here*>").

Students may also bring up the point that the security requirement won't necessarily stay Done once it's Done and might be better off as part of the Definition of Done. Accept this as an alternative problem/solution.

2 marks for finding problems (1 per problem, deduct 0.5 for vagueness, if more than two suggested then pick the best two). 2 marks for alternative phrasing – 1 per problem properly addressed. Nothing for fixing non-problems.

Question 2. (6 + 4 = 10 marks)

*This question is about **Scrum practices**.*

a) In typical scrum projects, sprint planning sessions are conducted at the start of each sprint so that the team will have a prioritized and estimated sprint backlog. One way to estimate tasks is to use Planning Poker.

When using Planning Poker, we usually give every member of the scrum team input into the estimation process. But that might mean that team members end up estimating tasks that are outside their experience – for example, a programmer might have to try to estimate how long a designer's job is going to take. Some critics claim that this is totally unrealistic. What is your stand on this criticism, and why? Write about half a page. **(6 marks)**

Possible pros:

- it's easier to estimate tasks that you are familiar with
- might make estimation quicker
- might allow designers to concentrate on design without interference from rest of team

Possible cons:

- this attitude tends to encourage siloing
- everybody is supposed to become at least a little familiar with everyone else's discipline
- what if "designer" story points end up diverging from "programmer" story points – could lose the ability to predict velocity
- the idea of an Agile team having a dedicated "designer" is a bit odd in itself

6 marks: well-reasoned argument that provides *at least three* pros and/or cons – note that the above lists are not exhaustive, accept any that make sense and are reasonable – and does not mischaracterize either Scrum or Planning Poker.

4 marks: argument that is waffly and/or underevidenced, or raises one or more points that are poor, but overall indicates a good grasp of Scrum **and** Agile estimation.

2 marks: says something correct and relevant about Scrum and/or Agile estimation.

0 marks: says nothing that is both correct and relevant about Scrum or Agile estimation.

Split the difference if torn between two levels. Deduct half a mark if vague.

b) What aspects of student projects make Scrum harder to manage compared to projects undertaken by typical industry teams? Write up to half a page. **(4 marks)**

Reasonable points include:

- Scrum recommends that people work on the project full-time and this is not typically possible in a university course
- Underperforming team members can't be sacked, and can quit without giving notice
- Teams might have no say in recruitment
- Teams don't get paid/incentive structure is very different
- Developers are inexperienced, which makes many aspects of development difficult
- Scrum Masters are inexperienced at facilitating/inexperienced with Scrum
- No need to support project after it is completed
- Usually, no need to deploy project

Many, many other points are possible.

4 marks: *at least three* distinct points raised and insightfully commented upon (or two points if the analysis is exceptionally good)
2 marks: at least two reasonable distinct points raised, with some attempt at analysis
0 marks: equivalent to no answer, or points are irrelevant or wrong

If torn between two categories, split the difference.

Question 3. (4 + 3 + 2 + 3 = 12 marks)

*This question is about **project inception**.*

Here is an idea for a product:

Rovr is a service that helps dog owners find people to provide services for their pets. You can think of it as "Uber for pets".

Service providers register themselves via a special app, verify their identity and contact details, and select which services they provide: bathing, clipping, dog walking, pet-sitting, etc. Pet owners can use the Rovr website or app to request a service. Rovr will offer the job to appropriate service providers in the local area. After the service has been provided, the owner gets to rate the provider's service on a scale of 1-5, and the provider can rate both the owner and the dog – the system charges an extra fee for handling dogs that are known to bite.

All transactions are logged and audited once a month to verify that service providers are being paid on a timely basis. A small team of technical staff at head office will manage these logs as well as maintaining the servers and responding to tech support requests.

a) Who should be invited to a story-writing workshop for this project, and why? Write about half a page. **(4 marks)**

- Candidate service providers, to talk about the "special app"
- Candidate pet owners, to talk about the "website or app"
- Staff members who will conduct the audits, to talk about how they should access required information
- The dev team, to get a better understanding of the requirements and user priorities
- The Product Owner
- The Scrum Master, to facilitate the meeting
- Technical staff, to talk about logging and server maintenance.

For full marks, answer must mention and justify end users, dev team, and head office support/finance staff. Max three marks if any one of these categories is missing; max one if two are missing. Deduct half a mark if vague, and a full mark if justifications are missing.

Other stakeholders are possible.

b) Suggest at least three personas that could be used in this workshop. **(3 marks)**

One mark per reasonable persona (drawn from stakeholders listed above).

Note: "personas" were defined in compulsory readings on story-writing workshops but not mentioned in lectures, so students who skimped on preparation won't know about them.

c) Consider the information that Rovr's system will need to store.

i. List at least *two* security-related risks related to the storage of this information. **(2 marks)**

Max one mark per risk. If they've given more than two, mark the two best. Risks include having a dog being falsely flagged as a biter, causing annoyance and expense for the owner; storing the addresses of people who are hiring pet-sitters and therefore might be away from home; identity theft risks arising from personally-identifiable information; etc.

ii. Suggest a mitigation strategy for at least one of these risks. **(3 marks)**
3 marks: strategy is feasible **and** likely to help
2 marks: strategy is feasible **or** likely to help
1 mark: strategy appears relevant to the nominated risk

Question 4. (4 + 4 = 8 marks)

*This question is about **Agile and traditional process models**.*

a) What problems with traditional processes do Agile process models help to solve? Write about half a page. **(4 marks)**

4 marks: At least two genuine and pressing problems, clearly articulated

2 marks: One pressing problem, clearly articulated, or two that have problems

0 marks: nothing relevant or correct

Problems you might see include:

- Anything relevant to the "software crisis": going over time/budget **due to documentation/management overheads**
- Inflexibility of requirements/unresponsiveness to client needs and wishes
- Inability to deal well with uncertainty in the business environment
- …many others

b) Give an example of a situation in which an Agile approach would *not* be suitable. Explain why. Write about half a page. **(4 marks)**

Similar to part a: need at least two characteristics for full marks. Need to both describe/characterize the situation **and** provide an explanation for full marks. See Week 12 lecture notes for example characteristics. You can afford to be picky; this is straight from notes on an open book exam.

Question 5. (4 + 4 + 6 = 14 marks)

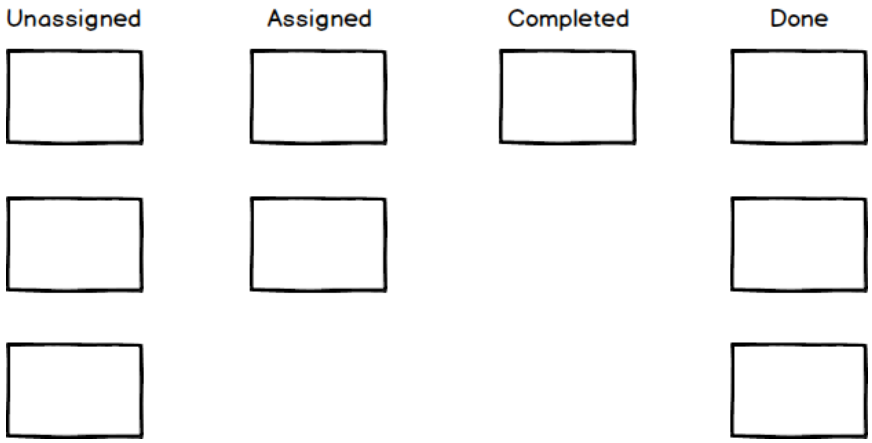
*This question is about **Agile practices**.*

A team is developing web services and web applications. Five years ago, the team used Scrum as documented in the Scrum Primer. Since then, their methodology has changed.

The team no longer uses user stories. Instead, requirements are managed by tasks. The Product Owner sits where she can see the master task list and notifies the Scrum Master if she sees anything that doesn't seem to be adding value from her perspective, and the Scrum Master investigates.

A master task list is kept in an online system. The task list is displayed at all times on a screen in the office that is visible from all team members' workstations, and is also accessible to team members when they are working offsite or from home.

Here is an image to show you the structure of this task list (note that this image is too small to let you read task details, but they would be easily readable on the screen.)



Whenever a team member is looking for something to do, they can either develop or QA. If they want to develop, they assign themselves a task from the "unassigned" column by putting their name on it and moving it to the Assigned column. When they feel it's finished, and it passes all unit tests, they move it to the Completed column. Team members who want to do some QA take a task from the Completed column, informally review the code, verify that it passes all automated tests and the acceptance tests as documented on the card, then move it to the Done column.

The team no longer conducts sprint planning or backlog grooming. Instead, team members do their own estimation at the time that they assign tasks to themselves. The Scrum Master and Product Owner check on an informal basis that the task list looks okay.

Instead of running a daily scrum, the Scrum Master makes sure to communicate one-on-one with each team member every day or two to informally check that everything is running smoothly.

CONTINUED ON NEXT PAGE…

Sprint reviews are no longer conducted because the team no longer feels that they are necessary. Instead, software is deployed to a staging server as soon as it passes unit tests and is manually pushed to production once it has passed user acceptance tests. This means that they are shipping new features much more often than they used to. The team does make sure to conduct a retrospective every two or three weeks to consider how their process can be made more efficient and effective.

a) Is this still Scrum? Why/why not? Write up to half a page. **(4 marks)**

No. (1 mark)
The "Product Owner" and "Scrum Master" have kept their titles but almost all the Scrum practices that distinguish it from other methodologies have been dropped: timeboxed iterations, sprint planning, sprint reviews, backlog grooming, Daily Scrum.

1 mark for "no, it's not Scrum". (No marks for saying that it's Scrum; it clearly isn't.)

3 marks for argument:

- 3: Cites evidence from question, clear and correct characterization of Scrum, explicit comparison that supports the answer
- 2: Answer demonstrates a shaky grasp of what Scrum is, but heads in the right direction and supports the answer
- 1: Some attempt to provide relevant evidence but answer does something silly such as assuming that Scrum and Agile are the same thing.

b) Is this still Agile? Why/why not? Write up to half a page. **(4 marks)**

Yes (1 mark) – it still follows the Agile Manifesto and the 12 Agile Principles, so it's Agile. In fact, the team is adhering more closely to these than previously – Scrum is pretty bureaucratic, as Agile methodologies go.

3 marks for argument, broken up as for part a)

The process described is actually Lean, but students don't have to say this. (I would be happy to buy a beverage for any student who recognized it though!)

c) Have the changes to this process increased or decreased the risks involved? What are the tradeoffs (i.e. costs and benefits)? Write up to a page. **(6 marks)**

The risks are definitely increased, but the overheads (e.g. in terms of meetings) have decreased and the team is releasing more frequently. 6 marks for addressing both pros and cons clearly; 4 for a reasonable but one-sided attempt, or for unclear, specious, or slightly ill-founded claims; 2 marks for an answer with at least one reasonable point.

Question 6. (2 + 2 + 4 = 8 marks)

*This question is about **team structure**.*

a) An Agile team may have as many developers as you like, but should only ever have one Product Owner. What risks arise if a team decides to appoint two Product Owners? Write about a paragraph. **(2 marks)**

If the Product Owners disagree, the project can become derailed in several different ways: differences in ideas about functionality, differences in prioritization, decrease in morale due to conflict, frequent miscommunication. Full marks for any one of these written up and thought through in detail, or two written up with less detail; 1 for one written up with little insight or detail.

b) What is *siloing*, and why is it considered a bad thing? Write a couple of sentences. **(2 marks)**

Siloing is the tendency for individuals or groups to take complete responsibility for a knowledge or skill domain, so that nobody else has access to that knowledge or skill set. It may also involve siloed individuals lacking knowledge or skills outside their silo.

It is bad because the siloed skillset may become a blocker if that silo is overloaded; on the other hand, if the silo is underloaded, its members may find themselves at a loose end, which wastes capacity. Other possibilities exist; pay if plausible.

1 mark for definition, 1 mark for at least one good reason.

c) Explain how DevOps approaches help overcome siloing, and how this improves outcomes for teams. Write about half a page. **(4 marks)**

2 marks for overcoming siloing, 2 marks for making it clear how outcomes are improved.

DevOps approaches place developers and operations staff in the same team and encourage them to collaborate and/or share their skills, which means that Operations skills are no longer siloed within the Operations team.

This improves outcomes because it allows Operations staff to make use of Dev skills to enable more frequent and less risky releases via increased use of scripting, and because it enables Dev staff to better understand the Operations attitude to risk, resulting in better and more appropriate pre-release QA. (Either of these points would be sufficient on its own and others exist.)

Question 7. (6 marks)

*This question is about **risk management**.*

A team member has recently attended an Agile conference. During your team's retrospective, he makes this suggestion:

"At the conference, I learned about the concept of YAGNI: You Ain't Gonna Need It. That is, you shouldn't design more functionality into your software than specified by your requirements, because doing so is likely to need to wasting effort, and you can always conduct a refactor if the extra functionality turns out to be necessary later on. That makes sense, right? But then I got to thinking, we spend ages thinking about risks, and they hardly ever happen. It's impossible to know what will happen until after it's happened, so why don't we apply the YAGNI principle to risk management and not write a risk register until after each risk has happened?"

What do you think of this suggestion – do you agree or disagree? Give reasons. Write half a page to a page. **(6 marks)**

Well, *I* reckon it's a bit silly – YAGNI is great when applied to not overrunning your requirements, but the idea behind risk mitigation is that you're trying to prevent bad things from happening so that they don't impact on your project. There's no point in doing this after the risk has happened. Besides, after the risk has happened it isn't a risk any more.

Some students will comment on YAGNI and not address the question about risk management. No marks if the answer doesn't address risk.

The only way an "agreeing" answer can get full marks is with an exceptionally well-reasoned and novel argument – frankly, I can't think of one.

6 marks: opinion clearly stated and supported with arguments that indicate understanding of Agile development and the nature of risk management, especially risk mitigation (i.e. reducing the likelihood or impact of risks).

4 marks: arguments are relevant but don't really support conclusion well, or argument is thin or poorly constructed.

2 marks: says something relevant but has no conclusion, or large part of argument is missing.

END OF EXAM