COMP33711

Two hours

Appendix A is located at the back of the exam

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Agile Software Engineering

Date: <some date>

Time: <some time>

There are TWO compulsory questions.
Please answer each question in a SEPARATE answer book.

For full marks your answers should be concise as well as accurate.
Marks will be awarded for reasoning and method as well as being correct.

This is a CLOSED book examination.

The use of electronic calculators is NOT permitted.

[PTO]

Question 1

a) The Agile Manifesto describes four "agile values" that the agile community believe underlie efficient and effective software development practices. The text of the manifesto (from agilemanifesto.org) is given below:

> *"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> *Individuals and interactions over processes and tools.*
> *Working software over comprehensive documentation.*
> *Customer collaboration over contract negotiation.*
> *Responding to change over following a plan.*
>
> *That is, while there is value in the items on the right, we value the items on the left more."*

Each of the agile practices listed below are based on one or more of these values applied to a specific aspect of software development. For each practice, name one value that it is based on, and briefly explain how it promotes the item on the left of that value over the item on its right.

i) Task boards
ii) "Done done"
iii) Business-value-based release planning
iv) On-site customer (i.e., customer as full team member)
v) Pair programming (10 marks)

b) You are a member of a team using agile development to build software for a major global toy manufacture and delivery company. The aim is to computerise their current manual process, which in outline is as follows. Children send letters listing their toy preferences. As the single annual delivery date approaches, specialist workers predict the location of each child on the delivery date. Using a quantum variant of the travelling salesman algorithm and a predicted model of wormholes in space-time around the delivery date, other specialist workers plan a route for the delivery. Somewhat surprisingly, the company employs only one member of delivery staff. Based on the planned route, a toy-mass-allowance is calculated for each child and a selection of toys from the initial letters is made (modified according to reports from field workers monitoring incidents of good and bad behaviour throughout the year). Manufacture of the toys can now begin.

i) Identify at least four different roles taking part in this manual process (optionally providing appropriate role names based on any domain knowledge you may have). (4 marks)

Question 1 continues on next page

(Continued from previous page)

ii) Using the Connextra story template presented in lectures, write 3 contrasting stories based on the manual process described above. At least one of the stories should be an epic, and at least one other should be small-scale enough to be suitable for implementation in a single two-week iteration. In your answer, indicate clearly which story is the epic and which the single iteration story. (6 marks)

iii) During agile planning, two members of the team (one senior and one junior programmer) give very different estimations regarding a given task. Choose a user story you gave in answer to sub question b) ii) and describe two possible explanations that these team members might have given. Finally, based on your explanations, propose a middle solution that can satisfy both sides giving the appropriate justification. (5 marks)

[PTO]

Question 2

a) You have been asked to look at some failing agile development teams, to diagnose the problems they are encountering. For each of the following scenarios, describe one potential root cause of the problem and one potential action to improve team progress.

i) The team consists of 3 developers, 2 testers and a customer representative. After a couple of iterations, the testers are found to be spending time writing tests for a couple of other (non-agile) teams. When questioned, they said that there was very little for them to do on the agile project and they were trying to make good use of their time.

ii) An organisation has decided to pilot the use of agile methods. Some developers with agile experience were employed, and a team was formed with a mixture of these new employees and some existing employees. After a few weeks, the existing employees complain to their manager that a lot of time is being wasted on writing unnecessary test code, pushing the code coverage levels up. These employees are worried that the testing team will have no work to do, when the project is passed on to them.

iii) A new agile team is formed to undertake a project for a customer. The team apparently gets off to a good start. A task board full of stories is created, and the team immediately begins to create lots of well-tested code. The customer representative is initially very happy, but soon starts to feel side-lined. When the team is coding, they keep their heads down and don't have much time to speak with him. When he does finally see a demo, the code that is being produced doesn't seem to match up with the vision on the post-it notes.

(6 marks)

b) You are a member of an agile team tasked with implementing the following user story for an e-commerce site:

> As a member of marketing staff, I want to offer discounts on orders from repeat and bulk-buying customers, so that we can encourage customer loyalty.

In conversation with the customer about this story, you discover that the marketing team wishes to offer the following discounts to customers who are members of the company's loyalty scheme:

- Free shipping on orders over £30 (not including shipping costs)
- Cheapest item free when 3 or more items are ordered

Question 2 continues on next page

(Continued from previous page)

Write a specification for this feature as a set of Cucumber examples (scenarios) using the Given-When-Then format. The examples should clearly resolve any ambiguities in the English description of the feature given above. (8 marks)

c) Write the step definition methods needed to enable *one* of the two discounts each of the steps in the Cucumber scenarios in your answer to part b) of this question, sketch out the Java step definitions that would be needed to make the scenarios executable. You should assume that no production code exists at this stage, so all production classes should be designed by programming-by-wishful thinking. Minor deviations from correct Java syntax will not be penalised provided the intention of the code is clear. (8 marks)

d) You have just joined an agile team as a developer. On your first day, you find the team embarking on a new high-profile project and in the middle of an intense debate as to whether the team should adopt TDD for the new project or not. Some of the existing developers have experience with TDD and some do not. Those in the anti-TDD camp say that the project is too important to risk experimenting with a difficult agile practice like TDD for the first time. Those in the pro-TDD camp say that the importance of the project is the reason for adopting TDD – it will help the team get the difficult business logic implemented correctly and avoid embarrassing bugs later.

Choose a side in the debate and give the argument you would make to your team, in support of this view. You may specify additional details about the project or the team to backup your argument, should you wish. (For example, you might specify more details about the urgency of the project, the size and make-up of the team, etc.) (3 marks)

END OF EXAMINATION

Marking Scheme for Sample COMP33711 Exam Paper

January 2018

Suzanne M. Embury
Christos Kotselidis

The exam format for this course unit has changed for this year, and now consists of just 2 compulsory questions, both worth 25 marks. The first question covers material on agile processes (roughly the material covered in weeks 1 to 5), while the second covers material on technical agile practices (roughly the material covered in week 7 to 12).

You should answer each question in a separate answer book. This is because each question is marked by a different person. Using separate answer books for each question allows us to mark the exam in parallel.

The sample exam follows the same structure, so you can get an idea of what to expect on the day.

This marking scheme describes how we would mark the answers to the sample exam, as well as providing model answers.

Question 1: set and marked by Christos Kotselidis

a) This question aims to test your understanding of both the basic agile values and some of the key practices we looked at in the course unit, by asking you to match up the practices against the value.

For each practice, I'll award 1 mark if it is matched with a plausible value, and one or more further marks for a good justification for the match, up to a maximum of 10 for the whole question. Good justifications will demonstrate a deep understanding of the concepts underlying both the value and the practice. If the same point is repeatedly made with regard to two or more of the practices, then that point will only be marked once.

For example, if a student matched every practice against value a), on the grounds that all the practices were carried out by people interacting, then than point would earn 1 mark the first time it was made, and 0 marks afterwards.

If no justification is given, just a value, then I'll have to make a judgement call as to whether I think the value is self-evidently and obviously linked to the practice. Please help me take the randomness this inevitably introduces out of my marking by always remembering to provide a justification.

Model answers are given below. Note that is it possible to match the practices plausibly against other values than those noted here. Many different answers could earn full marks if sensibly justified.

i) The task board practice matches with value a). In conventional teams, complex planning tools and practices are often used, which are often accessed electronically through complex authorisations. The planning tool or process becomes more important than the individuals on the team. The task board, on the other hand, is designed to support productive interactions between team members: to focus on the decisions themselves, not on how those decisions are recorded.

ii) The practice of assessing when stories are "done done" (as opposed to just "done") matches with value b). In this practice, a story is considered complete only when working software can be run in the deployment environment. This is the key test, and not whether any documents or checklists have been completed.

iii) This practice matches with value d). Business-value-based release planning involves planning releases so that high business value stories are delivered early, and in sensible combinations that together deliver business value. If the needs of the customer changes (so that the notion of what is high business value changes) then this can be reflected by changing the position of the stories in the release plan. The practice is lightweight, so the overheads of change are low.

iv) The on-site customer practice matches value c). In this practice, the customer sits with the team, to help it make decisions about what to build and what not to build. There is no contract to bind the two parties, and instead the customer is expected to work with the team as an equal team member, to make sure that useful software is built.

v) Pair programming matches with value a). It ensures high quality code by creating a situation in which two developers can interact continually, to both find problems and correct them quickly and informally. This is in contrast with the heavyweight code quality processes used on conventional projects (such as formal code reviews).

I've written the answers out fully here, so that they are comprehensible in this form. In the exam, you wouldn't need to write out the practice name and could just say:

i) Value a) because…

b) i) Roles clearly implied by the scenario are: location predictor, route planner, toy maker, toy deliverer, field worker. Toy manufacturing planner might also be considered to be implied, and child can be considered, if it is clear that there is some way in which they access the software, or otherwise have some major impact on business value.

1 mark to be awarded per sensible role with a satisfactory explanation. Just naming the role will award you 0.5 marks.

ii) 1 mark for each story that is clearly supported by the description of the scenario and that fits its description as either an epic or a single iteration story, plus an additional 1 mark for stories that correctly use the basic template, up to a total of 6 marks. Examples of acceptable stories are:

- As a toy deliverer, I want to know the location of the child I should be visiting next, plus my scheduled arrival time, so that I can **ensure** that every child on the route is visited, by the delivery deadline.
  (A **single iteration** story.)

- As a location predictor, I want to simulate the movements of groups of children based on a model I enter, so that I can **quickly** estimate their probable location on the upcoming delivery date.
  (Note the use of "quickly" in the last part of the story: the goal becomes a "next step" function without this. This story sounds like an **epic** to me.)

- As a location predictor, I want to be warned if children deviate from their predicted movement pattern in the weeks coming up to the delivery date, so that I can **maximise the accuracy** of my predictions for the final delivery.
  (Note: this is another story where it is very tempting to just put the next step in the process as the goal. You need to push through to the actual business value to get it right.)

- As a field worker, I want to submit reports of good/bad behaviour incidents, so that **fair** toy selections can be made for the final delivery.

Obviously, candidates would only need to provide 3 user stories in their own answer to the exam. We give 4 stories here only to increase the number of examples available to students for revision.

iii) You can pick a user story from your answer to b) ii) or present a new adequate one. For example, let's use the following user story:

*As a toy deliverer, I want to know the location of the child I should be visiting next, plus my scheduled arrival time, so that I can **ensure** that every child on the route is visited, by the delivery deadline. (A **single iteration** story.)*

Let's assume that during planning poker the senior developer rated it with 8 story points while the junior developer rated it with 2 story points. In this example, the more senior developer believes that this user story is more complicated in comparison to the junior developer.

Upon revealing the poker cards the senior developer argues the following:

> "I believe that this is a complicated user story because we have never worked with integrating Maps API in our applications and therefore we have no expertise. Furthermore, for that user story we have to combine the different addresses with our route planning software and dynamically adapt it in case we face delays during the delivery."

The junior developer argues the following:

> "I have previously worked with the API and it provides route planning functionalities. However, it does not dynamically re-calculate or re-calibrate the planned route upon delays."

We see from the answers above that:

- The senior developer did not know that a member of the team has previous expertise with this API.
- The junior developer did not think about the fact that they would need to dynamically re-calculate the planned route in case of delays.

A middle solution would be 5 story points, which will leverage the expertise of the junior programmer, but factor in that more work has to be done rather than just integrating the existing Maps API. (1 mark per sensible point given in the answer, or less in case not enough justification is provided).

Questions 2: set and marked by Suzanne M. Embury

a) 2 marks per scenario: 1 mark for a plausible problem diagnosis that fits the given facts, and 1 mark for proposing a corrective action that stands some chance of resolving the problem diagnosed. Below I give some examples of the kinds of point that could be made. Obviously, for just 1-2 marks, I would not expect people to write as much as I have written here.

i) An agile team where testers regularly don't have enough to do is definitely not operating correctly. As we saw in the lectures, testing is central to agile and should be taking place right from the time that the first story is gathered. Even before this, infrastructure for testing and continuous integration can be being assembled. In this case, it seems that the team is not gathering test cases as central part of their requirements gathering; or perhaps they are not automating test cases as they are extracted from the customer/team? Perhaps the testers on this team are operating to the traditional view that there is nothing for them to do until a full requirements specification has been written, or some code has been implemented. Perhaps they are not acting as one of the main points of interaction with the customer? The solution is to get them involved in story writing and test case automation right at the start of each iteration. They probably need some training on specification-by-example and/or automation of acceptance tests. And the team as a whole perhaps needs some coaching in the role of testing in agile projects, and also on whole-team-responsibility.

ii) This team appears to be working in an organisation that has traditionally separated testing out into a different function from development, and has siloed its staff based on quite narrow technical roles. Probably, the experienced agile developers are using test-driven techniques of some kind and are writing a lot of acceptance and unit tests as a means of discovering and documenting the requirements. But they have not taken the time to coach their fellow team members in the new approach. And the team seems not to have worked out explicitly how it will work with the dedicated testing teams the organisation owns. Such teams can still be useful for an agile team's work, but they should be used in addition to and not instead of the normal automated testing work of the agile team itself. The obvious "solution" is to provide more training for the existing employees, but this may not help them to change their mindset. More powerful approaches would be to focus on defining the team's own quality goals and how they will meet them (perhaps in an uncoming retrospective), or to spread expertise and awareness throughout the team through careful use of pair programming.

iii) Despite claiming to be agile, the team seems to be locked into a traditional "requirements gathering then implementation" approach. There was lots of interaction with the customer initially, but the team seems to think they don't

need to interact with the customer during coding. They are writing lots of tests, but they seem not to be checking that the tests they are writing actually do describe the behaviour that the customer will consider valuable. The result is inevitable: they are building the system they think the customer wants, instead of working continually with the customer to discover the best solution. Team training is always an obvious solution in these cases, but really the team needs to face up to the fact that they are building the wrong thing. One possible approach would be to shorten their iteration length, and to arrange highly visible showcases with multiple customer representatives at the end of each one. Introducing customer-facing testing techniques could also help, since the customer representative would be able to be more involved in checking that the tests match the vision for the product.

For full marks, the solution proposed should deal with any mindset problems in the team, as well as just addressing any missing skills. Generic answers saying that the team doesn't understand agile and needs to receive more training will only receive marks if backed up by justifications related to specific elements of the scenario in the question. Answers which could apply equally well to all three scenarios are unlikely to be specific enough to earn marks.

b) There are two types of discount mentioned and therefore we need to create two groups of scenarios to represent both of them. It is not necessary to provide full feature definitions in answer to this question (or any question in the actual exam on Cucumber). Just a sequence of Cucumber scenarios is all that is required, although candidates who do provide full features will not be penalised.

The shipping discount appears straightforward, and can be specified with just a few basic scenarios:

Given a customer who is a loyalty scheme member
And the customer places an order for products totalling £30.01
When the order is confirmed
Then the shipping costs for the order will be £0.00

Given a customer who is not a loyalty scheme member
And the customer places an order for products totalling £30.01
When the order is confirmed
Then the shipping costs for the order will be £3.00

Given a customer who is a loyalty scheme member
And the customer places an order for products totalling £30.00
When the order is confirmed
Then the shipping costs for the order will be £3.00

The question doesn't say what the shipping costs are or what they are based on. In this case, I assumed a simple single-cost model and made up a plausible value. In the actual exam, you should make similar decisions about aspects of the specification not

made explicit in the exam. Any plausible approach will be considered correct. If in doubt, you can add a note explaining what you were confused about and what you have done about it in your answer.

The scenarios needed to express the "cheapest item free" discount could be:

Given a customer who is not a loyalty scheme member
And the customer orders a book costing £10.00
And the customer orders a pen costing £3.50
And the customer orders a pad costing £5.00
When the order is confirmed
Then the order total before shipping will be £18.50

Given a customer who is a loyalty scheme member
And the customer orders a book costing £10.00
And the customer orders a pen costing £3.50
And the customer orders a pad costing £5.00
When the order is confirmed
Then the order total before shipping will be £15.00

Given a customer who is a loyalty scheme member
And the customer orders a book costing £10.00
And the customer orders a pad costing £5.00
When the order is confirmed
Then the order total before shipping will be £15.00

Given a customer who is a loyalty scheme member
And the customer orders a book costing £10.00
And the customer orders a pencil costing £1.50
And the customer orders a pen costing £3.50
And the customer orders a pad costing £5.00
When the order is confirmed
Then the order total before shipping will be £18.50

Given a customer who is a loyalty scheme member
And the customer orders a book costing £10.00
And the customer orders a pencil costing £3.50
And the customer orders a pen costing £3.50
When the order is confirmed
Then the order total before shipping will be £13.50

Notice that the wording here and in the previous stories makes clear that this discount is to be applied before the shipping costs are calculated. Other solutions which are consistent with the wording in the question will also be marked as correct.

Marks will be awarded as follows:

- Up to 2 marks for broadly correct use of the Given-When-Then structure. For full marks, the "when" step should describe an action taken by the user through the software by some mechanism, and should not be another pre-condition for the scenario. The "given" and "then" steps (including "and" and "but" clauses) should describe conditions on the state of the world/system.

- Up to 2 marks for writing scenarios that are compatible with the behaviour described in the question.

- Up to 2 marks for writing scenarios that cover the key aspects of the required behaviour. Here, specifically, there must be scenarios that show when the discounts are not applied, as well as scenarios showing when it is. The scenarios should address the key ambiguity of whether the shipping discount is applied after the "cheapest free" discount. However, it is not expected that students will specify all aspects of the behaviour. In general, it will be enough to give a scenario for the simplest version of the discount, a scenario when the discount is not applied, and a scenario showing some other aspect of the behaviour (like what happens when two items have the same cheapest cost). It is not necessary to give exactly the same set of scenarios as given above (or as many) for full marks.

- Up to 2 marks for writing scenarios that read well, and that use customer oriented language to describe the required behaviour in a way that non-technical customers will recognise. Technical terms such as "button", "dialogue box" and "menu" should be absent to earn these marks.

When answering questions that ask for Cucumber scenarios in the exam, you should feel free to make use of abbreviations and "ditto" marks to avoid having to rewrite scenarios out in full each time. For example, the version of the scenarios above is completely clear and acceptable:

Given a customer who is a loyalty scheme member
And the customer orders a book costing £10.00
And " " " a pen " £3.50
And " " " a pad " £5.00
When the order is confirmed
Then the order total before shipping will be £15.00

Longer words like "customer" can be abbreviated to "cust." in later scenarios provided the word has been used in full in some previous scenario and it can't be confused with some other key word used in earlier scenarios.

c) One step definition is needed for each distinct step used in the scenarios. This means that 4 step definition methods are needed for each discount option. Students are asked only to give step definitions for one of the two discounts, so 4 step definitions (two Given steps, a When step and a Then step) are required in the answer.

Marks should be awarded for the glue code as follows:

- 1 mark if all the steps in the selected scenarios are correctly matched by some glue code regular expression.
- 1 mark if all input values are matched and passed to the glue code correctly.
- 1 mark if the glue code for the When step invokes the production code and stores the result correctly.
- 1 mark if correct use is made of JUnit assertions in the glue code for the Then steps.

Marks should be awarded for evolutionary design/programming-by-wishful-thinking as follows:

- 1 mark for sensible domain class design (including names).
- 1 mark for sensible method design (including names, and the classes they are assumed to be the responsibility of)
- 1 mark if the classes are linked appropriately (which may be having no link at all, depending on the class design that is chosen)
- 1 mark for writing glue code that creates a sensible and complete fixture, including handling of mandatory attributes not mentioned in the scenarios.

The glue code needed for both discounts is given below (obviously, you would not need to give all this code in your answer, as step definitions for only one discount are requested in the question).

```
package uk.ac.manchester.cs.comp33711.sampleexam;

import static org.junit.Assert.assertFalse;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.core.Is.is;

import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;

public class SampleExamSteps {

   private Customer customer = new Customer("Freda", "Smith");
    private Order order = new Order(customer);

   @Given("^a customer who is a loyalty scheme member$")
   public void a_customer_who_is_a_loyalty_scheme_member() {
      customer.joinLoyaltySchemeMember();
   }

   @Given("^a customer who is not a loyalty scheme member$")
   public void a_customer_who_is_not_a_loyalty_scheme_member() {
      assertFalse(customer.isLoyaltySchemeMember());
   }

   @Given("^the customer places an order for products totaling
```

```
            £(\\d+)\\.(\\d+)$")
    public void the_customer_places_an_order_for_products_totalling(
                int pounds, int pence) {
        double cost = pounds + pence/100;
        Product product = new Product("Book", cost);
        order.add(product);
    }

    @Given("^the customer orders a (\\w+) costing £(\\d+)\\.(\\d+)$")
    public void the_customer_orders_something_costing(
                String productName, int pounds, int pence) {
        double cost = pounds + pence/100;
        Product product = new Product(productName, cost);
        order.add(product);
    }

    @When("^the order is confirmed$")
    public void the_order_is_confirmed() throws Throwable {
        order.confirm();
    }

    @Then("^the shipping costs for the order will be £(\\d+)\\.(\\d+)$")
    public void the_shipping_costs_for_the_order_will_be(
                int pounds, int pence) {
        assertThat(order.shippingCosts(), is(pounds + pence/100));
    }

    @Then("^the order total before shipping will be £(\\d+)\\.(\\d+)$")
    public void the_order_total_before_shipping_will_be(
                int pounds, int pence) {
        assertThat(order.totalBeforeShipping(), is(pounds + pence/100));
    }

}
```

d) This question aims to test candidates deep understanding of the strengths and weaknesses of TDD as a technique, as well as of the role of agile practices within agile teams. It is an open-ended question, and full marks can be obtained for answers that argue strongly for TDD as well as for answers that argue against the use of TDD in this situation.

Marks will be awarded for each unique valid point made in the answer.

Here is a sample answer that would earn full marks, and which is against the use of TDD:

> *I would argue against the use of TDD. I am assuming that the project is due to be delivered in a short time scale (in 3 months) and that at least half the team does not have TDD experience.*
> *TDD is known to reduce productivity in the short term and is a challenging technique to learn. With such a high proportion of the team new to TDD, even*

> *with pair programming it is going to take a while for the ideas to bed in, and 3 months is just not long enough for that. The team needs to be working well and producing code from day 1 of the project. I would suggest the team increases the amount of up-front testing it does on this project, but not to use full TDD.*

Here is a sample answer that would earn full marks, and which is in favour of the use of TDD.

> *I would argue for the use of TDD. I am assuming that the project is quite long (a year in total, though with early releases within that time scale of course, since this is an agile team). I am also assuming that the complex business logic referred to is really crucial to the project, and that getting it right is the team's biggest challenge in the project.*
> *With at least one experienced TDDer per developer pair, the technique can be picked up by the rest of the team in the timescales needed by the project. TDD will help the team implement the complex logic with confidence and will save the team time in debugging that logic later in the project, which will compensate for the time lost early on in learning the technique. The project will benefit in the longer term from the good test coverage that comes with TDD.*

Finally, here is a sample answer that would earn just 1 mark when marked generously, because of the vagueness of the reasoning and the lack of added information. Obviously, you don't get marks for restating what is said in the question, or for giving information not requested by the question (like the definition of TDD).

> *I would argue for the use of TDD. TDD is when you write lots of tests alongside the implementation of the production code, and use the red-green-green cycle. This project has complex business logic that is needed for TDD. The team has experience so it can learn the technique. It was able to learn other agile techniques, so should now try this one. The project is important so it should be well tested and TDD will help with this.*

Only this last sentence provides any new, relevant information, so 1 mark can be awarded, but no more.

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Agile Software Engineering

Practice Version of January 2017 Paper
Adapted for 2017/2018 Syllabus/Exam Format Changes

There are TWO sections. Please answer the COMPULSORY question in Section A, and any TWO questions from the FOUR questions provided in Section B.

For full marks your answers should be concise as well as accurate.
Marks will be awarded for reasoning and method as well as for correctness.

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

Question 2

a) Your team has been awarded a contract to deliver software for a regional chain of cinemas. The owner of the chain has a decade of experience of commissioning software for her own and other, larger organisations. She knows the pitfalls and takes a hands-on approach, having learnt that she needs to work closely with the development team, to make sure she gets software that meets her needs. She's worked with one other (Scrum-based) agile team, and was not impressed. They created lots of post-it notes, but the software that was eventually produced didn't seem to have much to do with what was written on them.

It's clear from the outset that you'll need to carefully tailor your agile processes to fit the needs and prejudices of your customer. Name two practices that you will adopt that will help your team to avoid the problems this customer has experienced in the past. Briefly explain why you think these practices will help this customer get the software she needs.
(4 marks)

b) You are working on the following user story:

> As a cinema sales manager, I want the ticket prices displayed as being available for each film to be appropriate for its showing type and classification, so that we always comply with government regulations on selling cinema tickets.

On the back of the index card bearing this story, the following conditions of satisfaction have been written:

- Film classifications available are U (unclassified), PG (parental guidance) and A (aged 18 and above).
- Showing types available are: "first run", "repeat run" and "weekend family".
- Basic ticket prices available are: adult (currently £10) and child (currently £5).
- Child tickets are only available for films with a U or PG classification.
- First run films are charged at 100% of the basic ticket price.
- Repeat run films are charged at 75% of the basic ticket price.
- Weekend family adult tickets are charged at 25% of the basic ticket price.
- Weekend family child tickets are charged at £1.00.
- Only films with a U classification can be shown as weekend family films.

i) Identify a simple happy path case from the information on the story card and write a Cucumber scenario describing one example of that case in action. (6 marks)

ii) Write 4 additional Cucumber scenarios, covering contrasting important cases based on the information given on the story card. (4 marks)

c) Give the glue code needed to automate one Given step, one When step and one Then step in the scenarios you wrote in your answer for question 2b).

Give a brief description of any domain objects that you assume the existence of (1-2 sentences), to allow the intent of your glue code to be understood.

You should assume that the scenarios will be run using the Cucumber-JVM, and therefore should write your glue code in Java. Minor syntactic errors will not be penalised, provided the overall intent of the glue code is clear. (6 marks)

d) You decide to use TDD to help you write the code that will make the scenarios you have written for this story pass. Select any of the domain classes you designed while writing the glue code for your answer to question 2c) as the starting point for coding. Give the state of the code for this domain class at each stage in the first TDD cycle you would carry out. Briefly explain what each stage involves, and why you have written the code you have written.

Minor syntactic errors in code will not be penalised, provided the overall intent of the glue code is clear. (5 marks)

END OF EXAMINATION

COMP23311 Practice Question 2

These answers was provided by students, and have been marked up by Suzanne as if they had been supplied as answers in the exam, as a revision aid to all students. I have added a little extra commentary in each case, beyond the level of feedback there is time to give when marking a large exam. Thanks to all the students involved for allowing their answers to be broadcast.

Answer from Anonymous Student #1

Q2.

a)

0/4

b)i)
Given a film classified as U
And the showing type is First Run
And the adult base ticket price is 10.0
And the child base ticket price is 5.0
When the ticket prices are refreshed
Then the adult ticket price is 10.0
And the child ticket price is 5.0

5/6

for this film?

✓ GWT broadly correct.

✓ Happy path.

✓ G. — is a precondition

✓ W — is an action

✓ T — is a postcond

b)ii)
Given a film classified as U
And the showing type is Repeat Run
And the adult base ticket price is 10.0
And the child base ticket price is 5.0
When the ticket prices are refreshed
Then the adult ticket price is 7.5
And the child ticket price is 3.75 ✓

Given a film classified as U
And the showing type is Weekend Family
And the adult base ticket price is 10.0
And the child base ticket price is 5.0
When the ticket prices are refreshed
Then the adult ticket price is 2.5
And the child ticket price is 1.0 ✓

Given a film classified as A
And the showing type is First Run
And the adult base ticket price is 10.0
When the ticket prices are refreshed
Then the adult ticket price is 10.0
And the child ticket price is not displayed. ✓

```
Given a film classified as PG
And the showing type is Repeat Run
And the adult base ticket price is 10.0
And the child base ticket price is 5.0
When the ticket prices are refreshed
Then the adult ticket price is 7.5
And the child ticket price is 3.75
```

c) Glue code

```
Film film = new Film("Super Combo");
Ticket adultTicket = new Ticket(film);
Ticket childTicket = new Ticket(film);
Board board = new Board();

@Given("^the adult base ticket price is (//d+)//.(//d+)$")
public void adult_base_ticket_price_is(int pounds, int pence)  {
    double basePrice = pounds + pence/100;
    adultTicket.setBasePrice(basePrice);
}

@Given("^the child base ticket price is (//d+)//.(//d+)$")
public void child_base_ticket_price_is(int pounds, int pence)  {
    double basePrice = pounds + pence/100;
    childTicket.setBasePrice(basePrice);
}

Given("^a film classified as (//w+)$")
public void film_classification_is(String classification) {
    film.setClassification(classification);
}

Given("^the showing type is (//w+)$")
public void film_showing_type_is(String showingType) {
    film.setShowingType(showingType);
}

When("^the story board is refreshed$")
public void story_board_is_refreshed() {
    board.add(film);
    board.refresh();
}

Then("^the adult ticket price is (//d+)//.(//d+)$")
public void adult_ticket_price_is(int pounds, int pence) {
    double expectedAmount = pounds + pence/100;
    assertEquals(expectedAmount, film.getAdultPrice());
}

Then("^the child ticket price is (//d+)//.(//d+)$")
public void adult_ticket_price_is(int pounds, int pence) {
    double expectedAmount = pounds + pence/100;
    assertEquals(expectedAmount, film.getChildPrice());
}

Then("^and the child price is not displayed$")
public void child_price_is_not_displayed() {
    assertFalse(board.isChildTicketDisplayed(film));
}
```

√g

Classification affects whether child tickets are displayed or not... all mixed up with price here.

good!

ticket or ticket type?

tweak name

? unclear how ticket class works...

gc : ✓ ½ ½

pow : ½ ½ ✓

Far too much code written.

Only needed to write 3 step defn. methods.

assumes 1 film shown

What checks that the right ticket price is displayed?

4/4

4/6

13/25

d) 0/5

Answer from Anonymous Student #2

a)

Would use ATDD (Acceptance Test Driven Development). Using specification by example to write Gherkin Tests, this means that the customer can write the specification in her own language. Therefore for the tests to pass, the software has to be built specifically for the specification she has asked for and will end up being exactly what she asked for. ✓½ Link to scenario

Could also use Short Iterations. We can assume that the team she worked with before did not use short iterations with frequent releases. By having frequent releases, we can get constant feedback from the customer and fail fast, as if the customer does not like what has been built then she will let us know and we can improve the system to fit her needs. ✓½

Paper prototyping. This is a cheap and quick way to make sure that before you start developing the customer can agree on what they want the software to do and look like. By using it with the customer before development we can get a clear finite idea of what she wants and we can make sure that the final result fits to her requirements. ✓½

3/4

b) Mark for best 2 awarded.

i) Given the film showing this week is an PG and it's on its first run
When ticket prices are refreshed on the board
Then the adult ticket price displayed is ~~100% of the basic adult ticket which is~~ £10 ✓✓✓½

4.5/6

ii) Given the film showing this week is an A
When ticket prices are refreshed on the board
Then a child ticket price won't be shown ✓

Given it is a weekend (Saturday or Sunday) and the film showing is a U
When ticket prices are refreshed on the board
Then the adult ticket price displayed is 25% of the basic adult ticket which is £2.50 ✓

Given the films showing is a PG and it's on a repeat run
When ticket prices are refreshed on the board
Then the adult ticket price displayed is 75% of the basic adult ticket which is £7.50 ✓

Given it's a weekend (Saturday or Sunday) and the film showing is a U
When ticket prices are refreshed on the board
Then the child ticket price displayed is £1

3/4

c) Assume that there is a java class called Film.java. This includes the methods about the film for example setCertification() and setShowingType().

Also assume that there is a java class called TicketBoard.java. This includes the methods that we need for the ticket board – refreshBoard().

And a java class called Ticket.java which has the method (getAdultPrice), this returns the ticket price for an adult that we would find on the board.

? Not clear what real world thing this refers to...

```
private Film film = new Film();
private TicketBoard ticketBoard = new TicketBoard();
private Ticket ticket = new Ticket();

@Given("film showing this week is PG and it's on its first run")
Public void film_showing_is_PG_and_on_first_run() throws Throwable {
        film.setCertificate("PG");
        film.setShowingType("first run");
}

@When("ticket prices are refreshed on the board")
Public void ticket_prices_refreshed_on_board() throws Throwable {
        ticketBoard.refresh();
}
@Then("adult ticket price displayed is 100% of the basic adult ticket which is £10")
Public void adult_price_displayed_is_10() throws Throwable {
        Int expectedAdultTicketPrice = 10;
        Int actualAdultTicketPrice = ticket.getAdultPrice();
        assertEquals(expectedAdultTicketPrice, actualAdultPrice);
}
```

d) For the Ticket.java class. TDD involves 3 stages:

1. Write a failing test – in this stage the Ticket.java class would have no code in it, therefore the test would fail.

2. Make test pass – Inside Ticket.java

```
Public int getAdultPrice() {
        Return( 10 );
}
```

This will make the test pass as we have hardcoded it to return the result that we are looking for.

3. Refactor the code - Import the Film.java class into Ticket.java

```
Public int getAdultPrice() {
        String showingType = film.getShowingType();
        If( showingType == "first run ")
                Int adultPrice = 10;
        Return(adultPrice);
}
```

This will make the test pass and has refactored the code to make it more readable and usable in the future, as we have used previous information that we set in the @Given step to determine the price of the ticket that we are looking for.

15/25

Note: I had to work much harder to see what practices you were suggesting here, because of desde layout.

Answer from Anonymous Student #3

a) One of the reasons the customer was not impressed with the previous team's agile practices, was because the user stories and task analysis done on the post-its did not match the final result. Paper prototyping or Wizard of Oz Testing would be a good strategy to adopt. As mentioned, the customer clearly enjoys the hands on process and by giving her prototypes of what to expect as well as letting her critique and add suggestions to the prototypes would mean there is no discrepancy between what she expects and what the developers assume. Another good practice to adopt would be acceptance testing. The customer is said to be comfortable with software jargon but even so, having a Gherkin acceptance tests in an abstract business language the customer understands would be helpful in solidifying the conditions of satisfaction for any user stories. We can assume the previous team did not work in short iterations and frequent releases. So including short iterations and incremental results following the fail fast technique would also be useful as she has a chance to verify if the intermediate product met her expectations.

practice, not strategy

½

Why can we? What in the scenario suggests this?

3/4 Taking marks from just 2 answers

b) i) Given an adult film is showing on its first run on a weekday
When the ticket prices are refreshed in the morning
Then the adult ticket price displayed is the same as the basic ticket price at £10

need to define this.

5/6

ii) ½ ½

? Behaves differently if refreshed in the afternoon?

A) Scenario 1:
Given there is a film only for adults (aged 18 and above) playing on a repeat run
When the display board is refreshed
Then the adult ticket prices are 75% of the original price

½

So the price displayed is?

B) Scenario 2:
Given that it is the weekend (Saturday/Sunday)
When the weekend family films are displayed
Then only U classified films are displayed

½

Could reuse steps from other scenarios.

C) Scenario 3:
Given there is a film classified as PG playing on its first run on a weekday
When the display board is refreshed
Then the adult ticket prices are £10 and child ticket prices are £5

D) Scenario 4:
Given there is a film only for adults (aged 18 and above)
When the display board is refreshed
There are no child prices displayed

3/4

c) Scenario 3

```
Public Class displayBoardPrices
{
        Ticket adultTicket = new Ticket();
        adultTicket.setTicketType("adult");
        Ticket childTicket = new Ticket();
        childTicket.setTicketType("child");

        Film film = new Film();

        @Given("Given there is a film classified as PG playing on its first run on a weekday")
        Public void filmClassifiedAsPGPlayingFirstRun () throws Throwable
        {
                film.setCertificate("PG");
                film.setShowingType("First");
        }
        @When("When the display board is refreshed")
        Public void refreshDisplayBoard() throws Throwable
        {
                displayBoard.refresh();
        }
        @Then("Then the adult ticket prices are £10")
        Public void adultTicketPricesBecome() throws Throwable
        {
                actualPrice = ticket.getAdultPrice(),
                assertEquals(10,  actualPrice);
        }
        @Then("the child ticket prices are £5")
        Public void childTicketPricesBecome() throws Throwable
        {
                actualPrice = childTicket.getChildPrice();
                assertEquals(5, actualPrice);
        }
}
```

refers to an actual ticket sold to an individual?

defined where?

No parameters extracted from the regex?

? How does this test what is displayed?

gc: ½ ½ g

pbw: ½ ½ ½

For this cucumber test example, I am assuming the existence of 3 classes; Film, Ticket and DisplayBoard. The Film class has methods to set and get its certificate (setCertificate, getCertificate) and also methods to set and get its showing type (setShowingType, getShowingType).
The Ticket class describes the type of ticket (adult or child) and has methods to set and get the ticket price (setTicketPrice, getTicketPrice).
The display board class describes the behaviour of the display board. It had one method - refresh, that when called will recalculate the conditions of all the films that are on the display board.

okay - useful, thanks - but why not call the class TicketType?

No connection between board & the film???

3.5/6

d) Three stages of test driven development

1) Create a failing test
In the ticket class for the previous example, in the first iteration, the class would be empty and thus would cause a failing test. ✗ Needs to be a failing unit test for TDD

2) Make failing test pass
To make the failing test pass, create method getPrice in the ticket class. Hardcode the value to make the failing test pass which in this case is

```
Public void getAdultPrice()
{
        Return 10;
}
```

½ g ½

3) Refactor the Code

- This involves adding logic to make the failing test pass rather than just hardcoding the expected results.
- Import Film;
- Public void getAdultPrice()
  {
  If (ticket.getTicketType() == 'Adult' && film.getShowingType('first')
  ticket.setTicketPrice("10")
  }

No - this is not the function of the refactoring step...

1/5

18.5/25