EDA397 / DIT191 Agile Development Processes
Exam

Thursday, Jun 2nd, 2016

Examiner
Eric Knauss +46 31 772 10 80

Contact person during exam
Magnus Ågren +46 736 47 24 91

Allowed tools / material

None except pen/pencil and eraser

General information

Numbers within parentheses show the maximal points awarded for each question. Maximal points can be given if:

- The answer is correct and correctly motivated.
- The presentation of the answer is readable and clear.
- The answer is given in English.

One sheet of paper may only contain parts of solutions belonging to one question.

Grading

The grades on this exam are based on your total score on the questions. For Chalmers students:

0 – 23 points:	Fail
24 – 35 points:	3
36 – 47 points:	4
48 – 60 points:	5

For GU students:

0 – 23 points:	Fail
24 – 47 points:	G (Pass)
48 – 60 points:	VG (Pass with distinction)

Results

Exam results will be made available through Ladok.

Review

The exam review will take place in Aug-18, 13:00 – 15:00, in Room J473.

General hint: Please be concise in your answers and make sure that you answer the question. Keep in mind that we always require you to motivate your answer and to demonstrate a good understanding of the subject matter. Points will be given for a) correctness of your answer, b) soundness of your argumentation, and c) general demonstration of knowledge.

The Agile Manifesto states the following **fundamental values**:

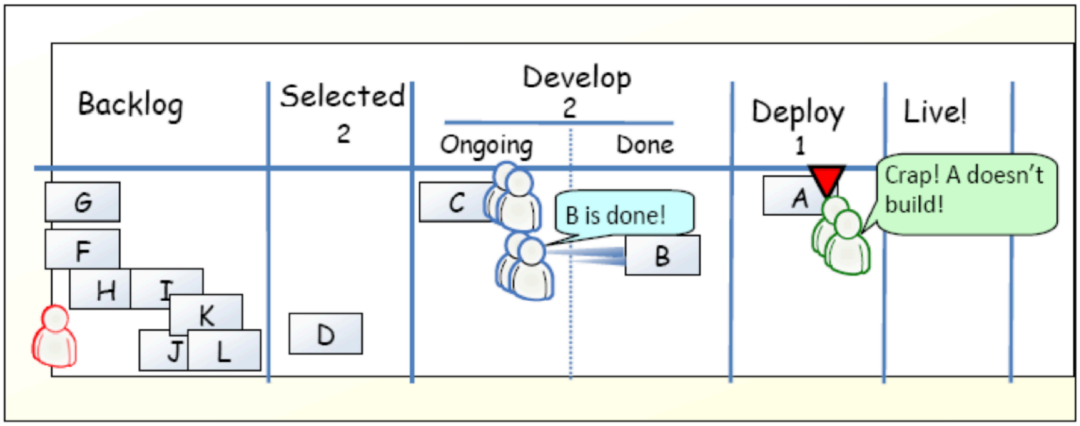
Individuals and interactions over processes and tools **Working software** over comprehensive documentation **Customer collaboration** over contract negotiation **Responding to change** over following a plan

Task 1: Test-driven dev. and automated tests **(12p; max 1pg)**

In this course, we have focused on the eXtreme Programming practice TestFirst.

- **a)** Describe the steps of the TestFirst practice and for each step give a short statement on why it is important. **(4p)**
- **b)** When is TestFirst difficult to apply, when is it easy to apply? **(4p)**
- **c)** State two other XP practices that support development of high quality software and explain how they complement TestFirst. **(4p)**

Task 2: Relate Kanban and Agile Development **(12p; max 1pg)**



a) With respect to the Figure above, what should each of the 4 actors (product owner, pair-1, pair-2, ops-team) do next? Where several valid alternatives exist, briefly outline them and decide which one should be taken. Please give a reason (4x 2p = **8p**).

b) With respect to the four fundamental values of the agile manifesto, discuss whether Kanban is or is not an agile method. For each fundamental value, support your argument with concrete examples of Kanban principles and core practices (**4p**).

Task 3: People and Communication Centric Dev. **(12p; max 1pg)**

- **a)** Pick and describe four Scrum practices. For each, discuss how as well as what kind of feedback and communication it supports. **(**4x 2.5p = **10p)**
- **b)** How do the Scrum practices from part a) and their feedback interact to support agile software development? **(2p)**

Task 4: Adapt Agile Methodology to Culture **(12p; max 1pg)**

Edward T. Hall distinguishes low-context cultures and high-context cultures. While the former relies on explicit and direct communication, the latter emphasizes interpersonal relationships and the exact meaning of communication often depends on the context of these relationships.

- **a)** Discuss the impact of low-context and high-context cultures on eXtreme Programming and support your argument with 3 examples of practices. **(**3x 2p = **6p)**
- **b)** Discuss the impact of low-context and high-context cultures on Scrum and support your argument with 3 examples of practices **(**3x 2p = **6p)**

Task 5: Transition to Agile and Agile Spirit **(12p; max 1 pg)**

Consider a project with problems:

> The project has a large requirements specification that is subject to frequent changes. The best designers do nothing but manage changes to this specification and their impact on other artifacts such as design specification, source code and test specification. Project members say things like "It's just too many documents. [...] Sure we need both user requirements specification and system requirements specification. But often, I change code and then go back to adjust the requirements." and "Why is the customer not working on the user req. spec? Are they confused by the many changes themselves?" and "System requirements specification? I know it is supposed to be useful. But currently I just try to keep it in sync with the unit tests we are writing." and "We probably should adjust the design document. It is outdated, but so far we seem to be all on the same page. It would be such a pain to bring it up to date!"

For this project, answer the following questions by highlighting differences between agile and plan-driven development, by referring to the agile manifesto, and by giving examples:

- **a)** *Short term:* How can you deal in an agile way with the fact that the project is late? Would you delay the release, deliver less functionality, or do you focus on delivering the functionality you and the customer agreed on, but cut costs in quality assurance? **(6p)**
- **b)** *Long term:* What should be done to make the project more agile (e.g. for future releases)? **(6p)**

Sketch of Answers

General comment: In this exam, we test for sufficient knowledge, but also for the student's ability to apply it in a structured argumentation and to transfer it to new situations. Thus, most of the Tasks do not have a single correct answer. In the following, we will sketch what we would judge as a good answer. Note, that we will give points even for incorrect answers as long as they are supported by a solid argumentation.

Answer to Task 1
a) Steps of test first:

- Write the test – there must be one to start with.
- Let the test fail – verifies that the failing case is possible.
- Implement until test passes – creates just the sought functionality.
- Refactor – refines the implementation beoynd bare passing.

b) Test first can be difficult to apply for example with:

- unclear requirements
- interactions with external software (API calls or IO)
- GUIs

Test first is easier to apply for example with:

- precise requirements
- computational logic, algorithmic code
- small concise fuctions
- stand alone, self-contained implementations

c)

Answer to Task 2
a) 1p is given per what each actor should do, 1p given per why each actor should do that.

Product owner selects one task – because of select limit.
Pair-1 keeps working on C – because they aren't done.
Pair-2 helps the ops-team, for example improving the infrastructure to alleviate the problem the ops-team experience – because the ops-team are blocked.
Ops-team continue working on A – because it's ongoing (not yet live).

b) Below are given the Kanban principles and core practices, marked +/- for whether they can be said to correspond to, or contradict, the agile manifesto. Note that these are our suggestions, see the general comment above.

Principles:
+ Start with shat you do now
+ Agree to pursue incremental/evolutionary change
– Respect the current processes, roles, responsibilities, and titles

+ Leadership on all levels

Core practices:
+ Visualize
+ Limit WIP
+/- Manage flow
– Make policies explicit
+ Implement feedback
+ Improve collaboratively, evolve experimentally

Answer to Task 3
a) Describe four Scrum practices (what the practice means =0,5p) and address both how the feedback (1p) and communication (1p) takes place within each practice.
Use information such as:

- …team member communicate by written…
- ….face-to-face communication with…
- …gives feedback on…. to...
- …illustrated by burn-down-chart…
- During stand-up meeting …discussed in team….

b) 1p per Scrum practice and its interaction with agile software development; 1p per explanation of how the practices interact with each other.

Answer to Task 4
a) Points are given for sound arguments describing the impact from culture on specific XP practices. Definitions of culture are accepted liberally; Hall's low- and high-context cultures serve as examples.

b) As for a) but with 3 practices from Scrum.

Answer to Task 5
General comment: It is important to read both parts first. Short term here was supposed to refer to the immediate future, potentially within this month. Long term refers to setting up a good, agile project culture. Independent from the parts a) and b), Task 5 asks for "*highlighting differences between agile and plan-driven development, by referring to the agile manifesto, and by giving examples*". Only few students have done this in both parts and we value that.

Part a) The correct answer is to reduce functionality. Quality cannot be sacrificed in agile, it would also backfire in the near future, when additional bug reports increase the time pressure even more. Delaying the release might be necessary in the very next days but should be discouraged. Most agile methods advocate time boxing. In addition, we should not delay customer feedback. An answer around those lines would receive **3 points**. An additional **1 point** would be given for highlighting differences between agile and plan-driven (e.g. that delaying the release would be the proper thing to do in plan-driven, since the delivery is defined in a contract). Similarly, if the answer referred to the agile manifesto (e.g. customer collaboration over contract negotiation to further showcase differences and make a point for reduced functionality), **1 point** was granted. If the answer was sufficiently specific, **1 point** was given for "example".

Part b) Since most students did already compare to plan-driven in part a), we did not give points for this here. Similarly, we felt that marking for the example would be unfair, since b) in nature is more abstract. As long as the answer was specific enough, we did not deduct points. We did however expect students to use the manifesto, which offers a reference for completeness. We gave **1.5 points** for each manifesto statement that was used, but deducted points, if only one side was used (e.g. working software or reduction of documentation, but not both), as well as when the proposed measures were not connected in a good way.