EDA397 / DIT191 Agile Development Processes
Exam

Thursday, Aug 24th, 2017

Examiner
Eric Knauss +46 31 772 10 80

Contact person during exam
Eric Knauss +46 31 772 10 80

Allowed tools / material

None except dictionary, pen/pencil, ruler, and eraser

General information

- Numbers within parentheses show the maximal points awarded for each question and the maximal number of pages that can be used.
- Please be concise in your answers and make sure that you answer the question. Observe the page limit. Text that significantly (=more than 1-3 lines) exceeds the space limit will be ignored. Consider striking an equal amount of earlier text if you want us to consider a late addition.
- Keep in mind that we always require you to motivate your answer and to demonstrate a good understanding of the subject matter. Maximal points will be given for:
    a) correctness of your answer,
    b) soundness of your argumentation,
    c) general demonstration of knowledge,
    d) the presentation of the answer is in English, readable, and clear.
- One sheet of paper may only contain parts of solutions belonging to one task.

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 Sep-15, 09:00 – 10:00, in Room J480 (please check course page for changes!).

Task 1: Multiple Choice (24p)

This part consists of multiple-choice problems. These problems consist of pairs of *propositions* and *reasons*. For each problem, you shall answer by using one of the following answers:

A Both the proposition and the reason are correct statements. In addition the reason explains the proposition in a correct way, i.e. the reason explains why the proposition is correct.

B Both the proposition and the reason are correct statements, but the reason does not explain the proposition.

C The proposition is a true statement, but the reason is false.

D The proposition is false, but the reason is a true statement.

E Both the proposition and the reason are false.

Correctly answered problems give **1 point**, while incorrect or missing answers give 0 points, regardless if you are partially correct in your answer!

	Proposition	*Reason*	*Answer A|B|C|D|E*
1)	In the case of stable requirements, agile projects tend to be more cost efficient than sequential development projects.	Sequential development projects struggle with change, since documents from previous phases need to be adjusted.	D
2)	According to [Mey14], the agile principle *"Working software is the primary measure of progress."* is a bad example of an agile principle.	According to [Mey14], a principle should be abstract, falsifiable, and prescriptive.	A
3)	An Onsite-Customer must possess extensive domain knowledge.	It is more important that the onsite-customers do not have to change their answers than to give an answer on the spot.	C
4)	Miniatures are good for new teams to get started in agile projects.	Miniatures provide an accurate and detailed depiction of the interplay of agile practices.	C
5)	The tendency in agile to rely only on scenarios for specifying requirements is problematic.	A user story describes a user goal and facilitates further discussion.	B
6)	Pair programming has generally a positive impact on cost efficiency.	Less resources are needed, since developers work in pairs (e.g. only one table, computer, …)	E
7)	The most important decisions in SCRUM are made during the daily SCRUM meeting.	Daily meetings avoid long/quiet crisis.	D
8)	It is hard to combine XP and SCRUM.	XP focusses more on development aspects, while SCRUM focusses on organizational aspects.	D
9)	Limiting work in progress contributes significantly to maximizing throughput.	Limited work in progress reduces multi-tasking and ensures that individual working steps are	A

		concluded as quickly as possible.	
10)	Limiting work in progress wastes resources.	Engineers will often be idle when waiting for bottlenecks, which cannot be easily removed.	E
11)	Iterations should be extended, if a small part of a user story could not be completed and tested.	In agile, it is strongly discouraged to sacrifice testing effort to make a deadline.	D
12)	Refactoring and TestFirst complement each other.	TestFirst provides regression tests to verify the success of a refactoring, refactoring can improve the testability of code.	A
13)	Agile teams can cover all testing needs through TestFirst and its automated Unit Tests.	TestFirst leads to a high line coverage of tests.	D
14)	The Lean Software Development Tool *Value-Stream Mapping* aims at increasing the profitability of agile projects.	The value-stream depicts how an individual cross-functional team provides customer value.	E
15)	It should be strictly avoided to develop features that are rarely used.	According to Standish group (2002), approximately 64% of features are never or only rarely used.	D
16)	Full utilization of developers provides no value to the overall value stream.	Queuing theory shows that delays through lack of steady rate of service significantly impact cycle time, leading to waste that outweighs slack of developers.	A
17)	Continuous integration requires a high degree of test automation.	Customer acceptance tests are crucial before releasing software.	B
18)	Continuous deployment can lead to slower feedback on integration problems in comparison to continuous integration.	The need for higher quality on the main branch increase feedback cycle time.	A
19)	Gradual rollout reduces the risk of high numbers of post-deployment problems.	Gradual rollout is a continuous integration practice that allows splitting large user stories over several iterations.	C
20)	Inherent difficulties to include developers in large-scale agile planning negatively impact efficiency of continuous deployment.	Continuous deployment implies local decisions of DevOps teams about customer visible features.	A
21)	LeSS encourages dedicated support for creating and maintaining software architecture.	Avoiding big-upfront analysis is a core value of agile.	D
22)	SAFe is ideal for small safety-critical projects.	SAFe is a framework to develop safety critical software in an agile way.	E
23)	Architecture helps to significantly reduce the risks and cost.	Architecture allows to discuss design decisions before implementation, when they are easy to adapt.	A
24)	Architecture negatively impacts the ability to be agile.	Architecture reduces the flexibility of cross-functional teams.	E

Sketch of solution

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.

Task 2: Contrast different agile methods **(12p; max 2pg)**

Based on the course book, we derived a revised set of agile principles. Your task is to compare two agile methods of your choice in the following subtasks.

a) Pick and describe two organizational and two technical principles **(4p)**

Make sure to give the correct number of principles and characterize them. Remember that there is some structure in the principles. E.g. if you choose "Develop minimal software", consider relating the different ways in that software can be minimal. If you choose test first, consider that its goal is to support to "Treat tests as a key resource"

b) Describe how the principles in a) are supported through concrete practices in Scrum. Alternatively, discuss a possible lack of support for a particular principle. **(4p)**

Aim for short, but clear answers. Avoid platitudes ("the sprint planning freezes requirements during iterations"), but explain ("In Scrum, the scope of a sprint is set during the sprint planning, thus freezing the requirements for a limited time. This manages change by having new/changed requirements wait for prioritization at the next sprint planning, thereby avoiding feature creep"). Generally, we look for correctness and sufficient depth of your answer as well as whether a good argument on how a practice supports a principle is presented.

c) Describe how the principles in a) are supported through concrete practices in Kanban. Alternatively, discuss a possible lack of support for a particular principle. **(4p)**

see answer to b.

Task 3: Difference in Leading Agile Teams **(12p; max 2pg)**

Cross-functional teams are an important concept in many agile methods.

a) Describe what a *Cross-functional team* is and how it relates to agile values and principles (**4p**).

A team that includes all necessary skills to deliver a feature, independently of other teams. Having the skills and the independence lets the team self-organize, and autonomously develop (and deliver) features, as well as quality assure them, without being stalled waiting for other teams to finish their work, or testers being available.

b) What roles are typically included in a cross-functional team? (**2p**)

Code, test, and UX. Scrum master/coach and product owner are usually **not** seen as part of the team, unless they do work from the backlog.

c) Discuss how cross-functional teams are visible in XP, Scrum, and Kanban (3*2 = **6p**).

XP includes the practices test first, requiring the team to be include testing skill, and collective code ownership, meaning team members need broad skills,

to deal with diverse parts of a complete software, for example GUI and database backend.
Scrum explicitly insists on teams being cross-functional, to the point of terming all team members "developers", regardless of the work they do.
Kanban, in contrast, doesn't prescribe cross-functional teams. Specialist teams are allowed, and the guiding principle is "respect the current process, roles, responsibilities, and titles".

Task 4: Small and Frequent Iterations. **(12p; max 2pg)**

a) Define continuous integration, continuous delivery, continuous deployment, and DevOps.(**4x1p**)

Continuous Integration: (The ability to) Integrate and test code every few hours, (1 day at the most)
Continuous Delivery: (The ability to) deliver software (to a customer) for installation at all times.
Continuous Deployment: (The ability to) install software in a running (customer) system at all times.
DevOps: Cross-functional feature teams (developers) work closely with operations teams to facilitate continuous deployment.

b) Name (**0.5p**) and explain (**1.5p**) 2 crucial issues to make continuous integration work.

- (Build and) Test Automation; required for high enough feedback speed. The executed test need to provide enough certainty that if they pass, the integrated software can be relied on.
- Version Control (System). Stable base to integrate into; for the test space to be manageable, the delta must be small. One change at the time.

Counter-check for "crucial": what would happen if that suggested wasn't in place?

c) How do continuous integration and continuous delivery relate to each other?

Continuous integration sustains a deliverable state of the software, although development is (continuously) ongoing. **(2p)**

Do they support or contradict each other?

Support, continuous integration is a prerequisite for continuous delivery(**0.5p**)

How? *Exemplifying capabilities gives points.*

Thus, feature requests and bug reports can be acted on, fixed, and swiftly (continuously) delivered. **(1.5p)**