EDA397 / DIT191 Agile Development Processes
Exam

Thursday, Jun 1st, 2017

Examiner

Eric Knauss +46 31 772 10 80

Contact person during exam

Magnus Ågren +46 736 47 24 91

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 Aug-18, 13:00 – 15:00, in Room J520 (please check course page for changes!).

Task 1: 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)**
b) Select one agile method and describe how the principles in a) are supported through concrete practices in that method **(4p)**
c) Select a second agile method and describe how the principles in a) are supported through concrete practices in that second method **(4p)**

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

User Stories are a central artefact in many agile methods.

a) Describe what a *User Story* is and what information it should contain (**4p**).
b) Describe the XP practice *Planning Game*. What is the basic idea, the players in this game, and their moves? (**4p**).
c) How do *User Story* and *Planning Game* support the goal to let a team self-organize? (**4p**)

Task 3: Small and Frequent Iterations. **(12p; max 2pg)**

a) Define continuous integration, continuous delivery, continuous deployment, and DevOps. **(4p)**
b) Name and explain 2 crucial issues to make continuous integration work. (**4p**)
c) How do continuous integration and continuous delivery relate to each other? Do they support or contradict each other? How? **(4p)**

Task 4: No Single Method fits all **(12p; max 2pg)**

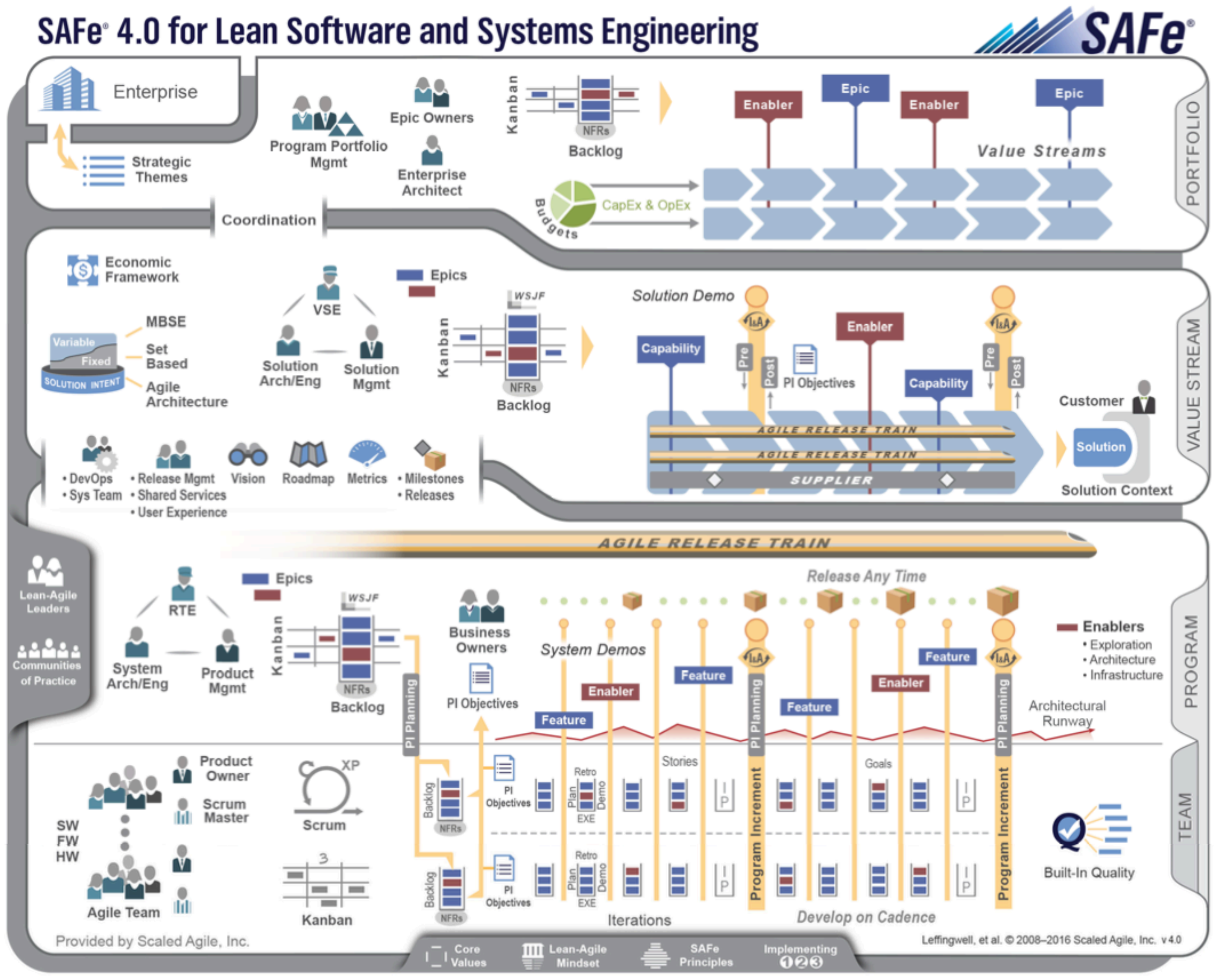
The role of architecture in agile methods is frequently disputed. There are at least two ways of creating an architecture within Scrum: (i) architecture is created based on special architecture backlog items in the Sprint backlog and (ii) architecture is defined in one or more dedicated architecture sprints before the real development starts.

a) Discuss pros and cons of each approach **(4p)**

Technical debt *is a concept in programming that reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution (Wikipedia).*

b) Give examples of agile practices (2 practices from Scrum and XP each) that can lead to technical debt. (**4p**)
c) Discuss: Does agile lead to more or less technical debt than plan-driven development? (**4p**)

Task 5: Scaling agile ideas to large organizations **(12pt; max 2 pg)**



The Figure above illustrates the Scaled Agile Framework (SAFe).

a) Discuss to which extent the agile value *Individuals and interactions over processes and tools* is addressed in SAFe from the perspective of the teams? Give two examples where it is visible and two where it is not. (**4p**)

b) Discuss to which extent the agile value *Customer collaboration over contract negotiation* is addressed in SAFe? (**4p**)

c) Discuss to which extent the agile value *Responding to change over following a plan* is addressed in SAFe, e.g. in the release train? (**4p**)

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 1:

a) 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) And c): Aim for short, but clear answers. Avoid platitudes (*"test first supports test first"),* but explain (*"In XP, the test first practice consists of writing automated unit tests and only then write just enough productive code to make the tests pass; test first has led to more general test-driven development (not limited to unit tests), which has then been raised to an agile principle"*). 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.

Task 2:

a) **Describe what a User Story is and what information it should contain (4p).**
- As a <**Role**> I want to <**feature**> in order to <**business value**>
- A **user story** is an informal, natural language description of one or more features of a software system, with a focus on **what** and **why**
- Written from **the perspective of an end user** or user of a system.
- Often written on index cards or post-it notes
- Depending on the project, user stories may be **written by various stakeholders** (not only the developers), including clients, users, managers or development team members.
- Usually only covers FR (and not NFR)
- Can indicate a **size, priority,** and a unique **identifier**

b) **Describe Planning game (basic idea, players, and moves) (4p).**
- Technique for **estimating**
- Usually played by **Customers** and **Developers**
- Actors seek an optimal compromise between **business value** and **development**
- Story points could be **ideal days, hours or other units** in which the team estimates.
- The game process is repeated **until consensus is achieved** or until the estimators decide that agile estimating and planning of a particular item need to be deferred **until additional information** can be acquired.
- Do not describe planning poker (only)

c) **How do US and PG support goal to let a team self-organize (4p)?**
Self-organizing issues to relate to, such as;

- The team **pull work for themselves** and don't wait for their leader to assign work.
- They manage their work (such as allocation, estimation, development, and refactoring) as a group
- Developers have **estimated the effort together** (compared to a top-down approach), and therefore the tasks can be equally shared within the group (can avoiding conflicts)
- The team estimate themselves how much work they can take on
- The team **communicate** more with each other, and their commitments are often linked to the project teams (instead of the manager)
- Important things for self-organizing team i.e. team empowerment, collaboration, motivation, continuity, and competence

Task 3: Small and Frequent Iterations. **(12p; max 2pg)**

a) Define continuous integration, continuous delivery, continuous deployment, and DevOps. **(4x1p)**
CI: (The ability to) Integrate and test code every few hours, (1 day at the most)
Cdel: (The ability to) deliver software (to a customer) for installation at all times.
Cdep: (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. (**4p**)
(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.
VCS. Stable base to integrate into; for the test space to be manageable, the delta must be small. One change at the time.
crucial; counter-check: "what would happen if that suggested wasn't in place"

c) How do continuous integration and continuous delivery relate to each other?
CI sustains a deliverable state of the sw, although development is (continuously) ongoing. (2p)
Do they support or contradict each other?
Support, CI prerequisite for Cdel (0.5p)
How? *Exemplifying capabilities gives points.*
Thus, feature requests and bug reports can be acted on, fixed, and swiftly (continuously) delivered. (1.5p) (4p)

Task 4:

Architecture is based on architectural backlogs in the sprint backlog **OR**
Architecture is defined in dedicated sprint/s before the real development starts.

a) **Describe Pros and Cons of each approach (4p).**

- An agile way to define an architecture, using an iterative lifecycle, allowing the architectural design to **tactically evolve gradually**, as the problem and the constraints. By being agile, architecture will **gradually emerge, out of bi-weekly refactorings**. This belief was amplified by a rather poorly worded principle #11 and cemented by the profuse amount of repeated mantras.
- Key architectural choices **cannot be easily retrofitted** on an existing system by means of simple refactoring.
- Much of the architectural decisions **have to be taken early**, although not all at once up front
- If well-defined requirements are available before start or not.
- Flexibility, easily adapt to changes
- In the best of worlds, we would like to have an agile process, leading to a flexible architecture.

b) Four examples of agile principles leading to TD (4p).

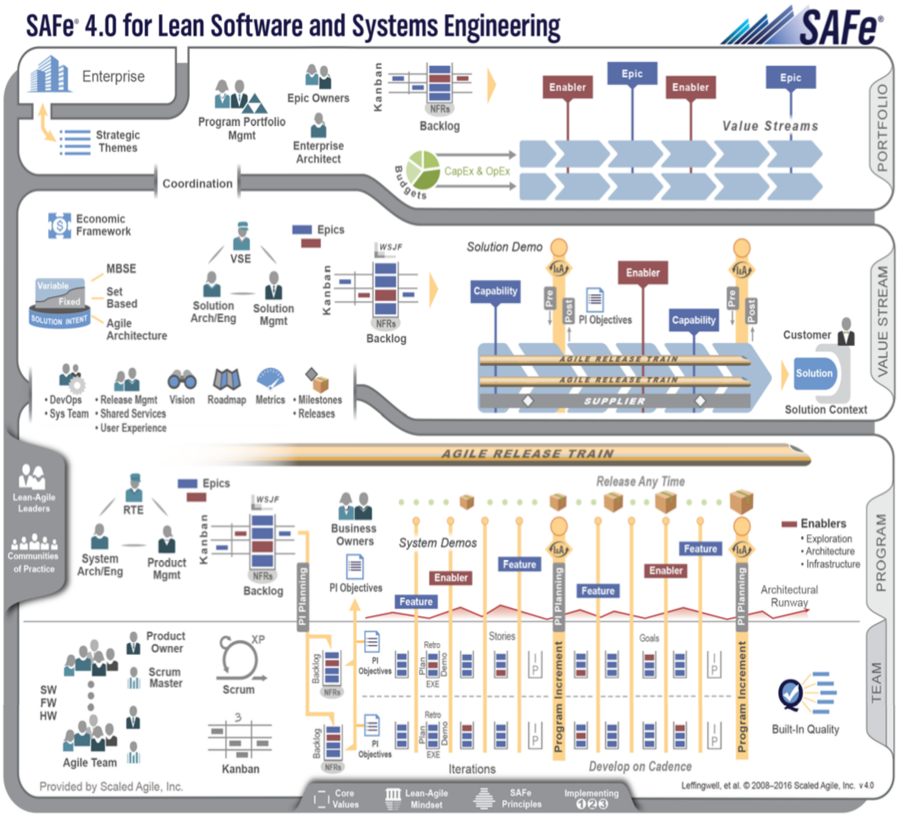
Some examples:

- Simply design – tradeoff in the architectural specification can lead to TD
- Burn-down charts can stress the developers to implement sub-optimal solutions to reach the time limit, which has to be refactored later.
- Writing stories with a user focus, can lead to an unclear requirement that can lead to sub-optimal solutions.
- When using the agile approach of formulating story or use cases, and the definition of "Done" is not clear, this can lead to requirement technical debt.

c) Discuss: Agile leads to more or less TD than plan-driven development (4p).

There is no right and wrong in this question, but we need to consider the level of **changing requirements**, **refactoring strategies** (iteratively) and also the agile focus on continuously **delivering new features** in every sprint.
An example could look like this "Because they use an iterative development process, many agile teams seem to believe that they are completely immune to technical debt. Although iterations offer the opportunity to reimburse debt in a timely fashion, the opposite often occurs. Developing and delivering very rapidly, with no time for proper design or to reflect on the longer term, and a lack of rigor or systematic testing (including automated testing) leads some agile projects into massive amounts of debt very rapidly. In fact, such debt can mount much more quickly than in any old-fashioned waterfall-like project. however, in the end, it's all a matter of choice: where time to market is essential, the debt might actually be a good investment, but it's imperative to remain aware of this debt and the increased friction it will impose on the development team, as Cunningham suggested" (taken from "Technical Debt: From Metaphor to Theory and Practice", by Philippe Kruchten)

Task 5: Scaling agile ideas to large organizations **(12pt; max 2 pg)**



The Figure above illustrates the Scaled Agile Framework (SAFe).

a) Discuss to which extent the agile value *Individuals and interactions over processes and tools* is addressed in SAFe from the perspective of the teams? Give two examples where it is visible and two where it is not. (**4p**)

+ Teams are cross-functional, and can choose how they work, as long as they can deliver in iterations.
+ Teams part of the planning.
- Large framework requiring adherence; prescribed control outside the team.
- High reliance on automation (and DevOps). Requires tool support (and that needed tools can operate in a CI env.)

b) Discuss to which extent the agile value *Customer collaboration over contract negotiation* is addressed in SAFe? (**4p**)

Always certain distance to customer in organizations of the size SAFe is aimed at.
Required relation to scale for full points on b.
Customer is the Business Owner during planning; part of planning.

c) Discuss to which extent the agile value *Responding to change over following a plan* is addressed in SAFe, e.g. in the release train? (**4p**)

SAFe retains the Scrum idea that it's fine to wait, if not for too long. If missing one train, just take the next, as long as trains go often.
Short commitment horizon, regular (re-)planning.
(ART is a combination of several Agile teams)
multi-level backlog(s) corresponds to a kind of plan